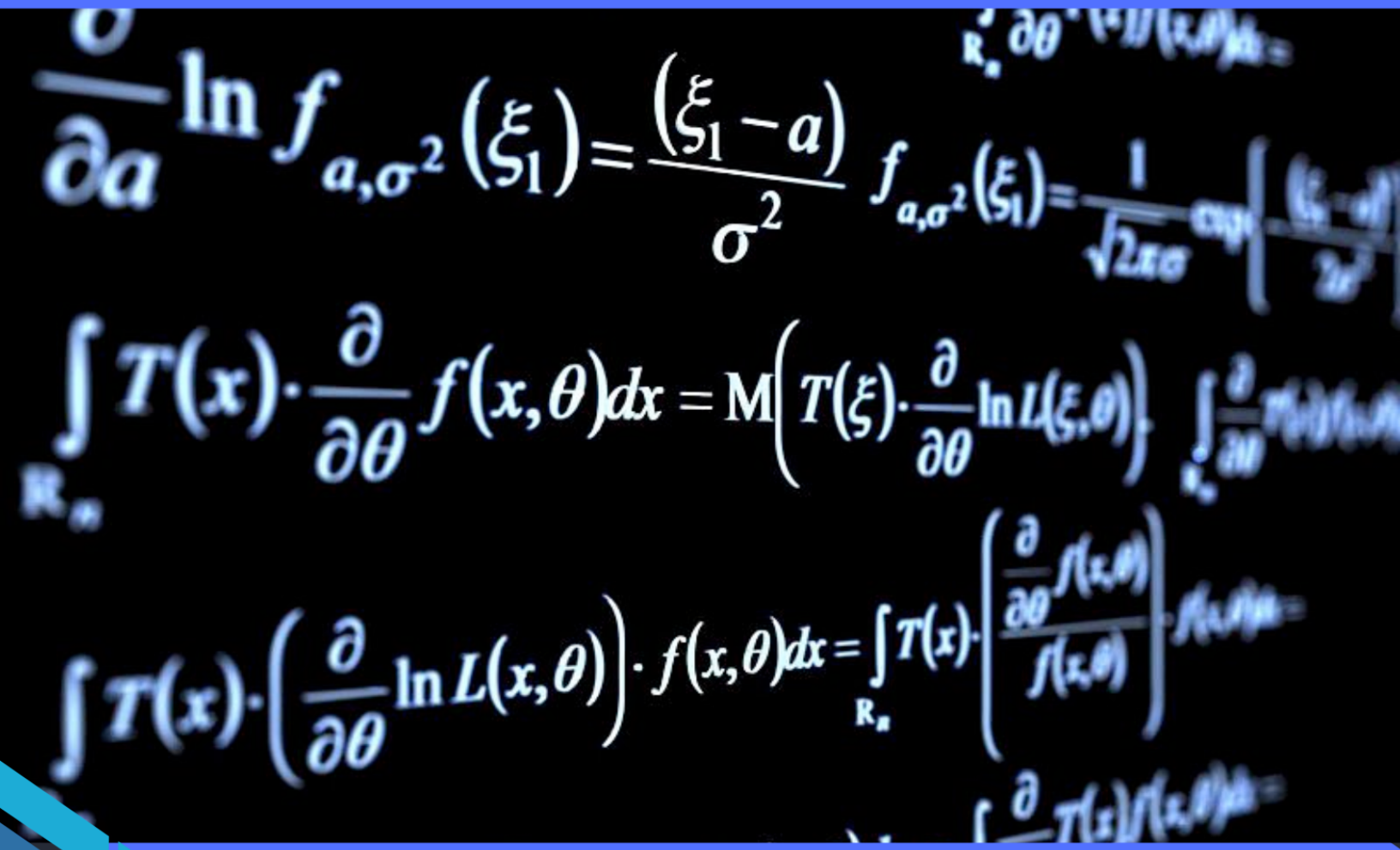
AN INTRODUCTION TO PYTHON JUPYTER NOTEBOOKS

FOR COLLEGE MATH TEACHERS



edited by

Paul Isihara, Claire Wagner, Peter Jantsch, and Thomas VanDrunen

An Introduction to Python Jupyter Notebooks for College Math Teachers

Paul Isihara, Claire Wagner, Peter Jantsch, and Thomas VanDrunen, Editors

Jul 26, 2024

Contents

1 PREFACE **2**
- 1.1 JMM 2023 . . . 3
- 1.2 JNB LAB: Patterns . . . 3
- 1.3 JNB Lab Solutions . . . 12

2 PRE-COLLEGE - Wheaton College Team **17**
- 2.1 Introduction . . . 17
- 2.2 Elementary Blackboard Problems . . . 18
- 2.3 Beginning Python Programming . . . 27
- 2.4 Glimpse of Chicago . . . 37
- 2.5 Arts in STEM (STEAM) . . . 48
- 2.6 Solution to Exercises . . . 65
- 2.7 JNB LAB: After-School Program Demo . . . 82
- 2.8 JNB Lab Solutions . . . 93

3 PYTHON PROGRAMMING GUIDE - Thomas VanDrunen, Yiheng Liang **98**
- 3.1 Introduction . . . 98
- 3.2 Introduction to Python . . . 98
- 3.3 Working with Data . . . 118
- 3.4 Solution to Exercises . . . 141
- 3.5 JNB Lab: Introduction to Python . . . 146
- 3.6 JNB Lab Solutions . . . 153

4 EXPLORATORY DATA ANALYSIS - Jonathan Zhu **155**
- 4.1 Introduction to Part I . . . 155
- 4.2 Data Investigation I . . . 156
- 4.3 Plotting and Statistical Testing I . . . 160
- 4.4 Solutions to Part I Exercises . . . 169
- 4.5 Introduction to Part II . . . 172
- 4.6 Data Investigation II . . . 175
- 4.7 Plotting and Statistical Testing II . . . 193
- 4.8 Making EDA Reports . . . 207
- 4.9 Solutions to Part II Exercises . . . 211
- 4.10 JNB Lab: Housing Equity Initiative . . . 214
- 4.11 JNB Lab Solutions . . . 223

5 PROBABILITY - Laura Gross, Yiheng Liang **231**

5.1 Introduction . . . 231
5.2 Random Numbers . . . 232
5.3 Probability . . . 235
5.4 Conditional Probability . . . 239
5.5 Probability Distributions . . . 242
5.6 Random Walks . . . 249
5.7 Agent-based Models . . . 253
5.8 Mathematical Games . . . 255
5.9 Solution to Exercises . . . 259
5.10 JNB Lab: Homicides in Chicago . . . 287
5.11 JNB Lab Solutions . . . 290

6 STATISTICAL INFERENCE - Peter Jantsch, Claire Wagner 294
6.1 Introduction . . . 294
6.2 Foundations of Statistical Inference . . . 296
6.3 Hypothesis Testing for Categorical Data . . . 306
6.4 Hypothesis Testing for Numerical Data . . . 320
6.5 Linear Regression . . . 332
6.6 Solutions to Exercises . . . 357
6.7 JNB Lab: The Framingham Heart Study . . . 373
6.8 JNB Lab Solutions . . . 375

7 APPLIED CALCULUS FOR DAILY LIFE - Wheaton College Team 377
7.1 Introduction . . . 377
7.2 Linear Systems and the Two-commodity Model . . . 378
7.3 Marginal and Average Cost . . . 380
7.4 Optimization and Object Design . . . 383
7.5 Optimization and Cobbs-Douglas Production . . . 387
7.6 Related Rates and Volumes . . . 388
7.7 Related Rates with Trig Functions and Football . . . 391
7.8 Probability Distributions and Drive Thrus . . . 396
7.9 Normal Distribution and Process Control . . . 403
7.10 Partial Derivatives and OLS Regression . . . 406
7.11 Area Between Curves and the Gini Index . . . 409
7.12 Integral Test and Income Streams . . . 412
7.13 Ordinary Differential Equations and Exponential Growth/Decay . . . 415
7.14 Solutions to Exercises . . . 419
7.15 JNB Lab: Calculus of Entropy in Daily Life and Monte Carlo Simulations . . . 437
7.16 JNB Lab Solutions . . . 448

8 CALCULUS - Inne Singgih 452
8.1 Introduction . . . 452
8.2 Functions . . . 453
8.3 Limits, Continuity, and Rates . . . 462
8.4 Derivatives . . . 469
8.5 Integrals . . . 482
8.6 Parametric Equations . . . 497
8.7 Sequences and Series . . . 511
8.8 Solution to Exercises . . . 523
8.9 JNB LAB: Calculus Animations . . . 538
8.10 JNB Lab Solutions . . . 544

9 LINEAR ALGEBRA - Soheil Anbouhi 549
9.1 Introduction . . . 549
9.2 Linear Systems . . . 550

9.3 Matrices and Determinants . . . 568
9.4 Linear Transformations . . . 595
9.5 Eigenvalues and Eigenvectors . . . 615
9.6 Orthogonality . . . 633
9.7 Solutions to Exercises . . . 661
9.8 JNB Lab: Linear Algebra . . . 696
9.9 JNB Lab Solutions . . . 699

10 LINEAR ALGEBRA AND OPTIMIZATION FOR DATA ANALYSIS - Wheaton College Team 714
10.1 Introduction . . . 714
10.2 OLS Linear Regression . . . 715
10.3 K-means Clustering . . . 720
10.4 Dimension Reduction by Principal Component Analysis . . . 725
10.5 Binary Classification of Labeled Data by Support Vector Machines . . . 732
10.6 Solution to Exercises . . . 741
10.7 JNB LAB: Linear Algebra for Data Analysis . . . 744
10.8 JNB Lab Solutions . . . 767

11 DIFFERENTIAL EQUATIONS - Rachel Petrik 778
11.1 Overview of Chapter . . . 778
11.2 Introduction to Differential Equations . . . 779
11.3 First-Order Differential Equations . . . 781
11.4 Second-Order Differential Equations . . . 793
11.5 Systems of First-Order Differential Equations . . . 801
11.6 Laplace Transforms . . . 823
11.7 Solutions to Exercises . . . 831
11.8 JNB Lab: Differential Equations . . . 852
11.9 JNB Lab Solutions . . . 858

12 DIFFERENTIAL EQUATIONS FOR THE BENEFIT OF SOCIETY - Wheaton College Team 880
12.1 Introduction . . . 880
12.2 Logistic Growth and COVID-19 . . . 881
12.3 The Basic SIR Model . . . 885
12.4 Cholera in Haiti . . . 888
12.5 CWS Model of Alzheimer's Disease . . . 891
12.6 Gravity Fed Water Delivery . . . 903
12.7 Earthquake Resistant Construction . . . 929
12.8 Solutions to Exercises . . . 943
12.9 JNB Lab: HIV-AIDS . . . 957
12.10 JNB Lab Solutions . . . 968

13 COMPLEX VARIABLES IN GROUNDWATER MODELING - Wheaton College Team 972
13.1 Setting the Scene . . . 972
13.2 Complex Analysis Background . . . 974
13.3 Idealized Groundwater Flow . . . 981
13.4 Contaminant Extraction Modeling . . . 988
13.5 Solutions to Exercises . . . 996
13.6 JNB LAB: Complex Potentials and Contaminant Flow . . . 998
13.7 JNB Lab Solutions . . . 1005

14 ADVANCED DATA ANALYSIS - Jonathan Zhu, Claire Wagner, Ying Li 1012
14.1 Introduction . . . 1012
14.2 Preprocessing . . . 1015
14.3 About Text Data . . . 1025
14.4 Document Embedding . . . 1037

14.5 Classification of Texts . . . 1041
14.6 Evaluating Classification . . . 1046
14.7 Applications . . . 1053
14.8 Solutions to Exercises: Sections 1 to 4 . . . 1061
14.9 Solutions to Exercises: Sections 5 to 7 . . . 1076
14.10 JNB Lab: United States Grant Data . . . 1096
14.11 JNB Lab Solutions . . . 1100

15 COMPLEX SYSTEMS - Wheaton College Team 1116
15.1 Introduction . . . 1116
15.2 Fundamental Concepts . . . 1119
15.3 Mathematical Concepts from Equilibrium Statistical Physics . . . 1128
15.4 Societal Applications . . . 1137
15.5 Solution to Exercises . . . 1150
15.6 JNB LAB: Complex Systems . . . 1155
15.7 JNB Lab Solutions . . . 1169

Note

This Jupyter Book is a PDF version of a project available on GitHub. We encourage readers to visit the online version in addition to using this PDF.

Some of the resources in this book may involve interactive elements or other dynamic content that cannot function fully in PDF format. In these cases, we have included notes suggesting that reader consult the online version for full functionality.

Additionally, for the sake of concision, some of the Python code has not been included in the PDF. Any such code can be viewed on the online version or downloaded from the GitHub repository.

CHAPTER 1

PREFACE

This book is an introduction to the use of Python Jupyter Notebooks (JNBs) for college math teachers and their students. The book is an evolving work in progress, but we hope this 1st Edition may already be quite useful to a college math teacher who is interested to use Jupyter Notebooks in their courses. Each section of the book has been written in the form of a Jupyter Notebook and can be downloaded using the download button at the top menu bar. In some cases, additional data files or image files may need to be downloaded as well. All the files are available in the src folder of the github repository https://github.com/timothyprojectGiG/JB_Math_Textbook

Each chapter contains a JNB lab with solutions. Experienced teachers can modify these labs and create new labs tailored to their courses. The chapters were written by different authors/authorship teams, and as such, vary in style. Each chapter essentially can be read independently of the others as long as one has the pre-requisite mathematical knowledge. Some effort has been made to standardize the chapter formats, and the index may help the reader find specific topics of interest. There is virtually no limit to what can be done with JNBs, and we hope this work serves as a useful starting point for teachers and students to enrich and enliven the study of mathematics using this tool.

The JNBs used to create this book contain special markdown code which are needed to produce features such as indexing, referencing, and highlight boxes in the Jupyter Book. Some editing of a downloaded JNB may may make it more suitable for classroom use.

To the First-Time Jupyter Notebook (JNB) User

Welcome to the wonderful world of Jupyter Notebooks (JNBs)!

In this book, we will be using JNBs as type of user-friendly way to input and execute codes in the Python programming language. JNBs also are useful to create documents which include photos, videos, slide presentations, as well as output to Python computer programs. In fact, this entire book is written using JNBs.

To get started, it will be very helpful if you consult with someone who is already using JNBs. If that is not possible, you will need to install Jupyter Notebooks using the free Anaconda distribution downloadable from www.anaconda.com. Next, skip to Section 2.3 Beginning Python Programming in the PRE-COLLEGE chapter (if you are using the online book, you can navigate using the left menu bar). Reading this section will hopefully help you install and run your first JNB. Then you can try to work through the rest of the PRE-COLLEGE chapter.

We wish you all the best!

This first edition would not be possible except for the contributions of a congenial team of faculty who worked together during Summer 2023 (acknowledged as authors of their contributed chapters), as well as a team of students (Jakob Hildebrandt, Gavin McClowry and Jonathan Zhu) who did a variety of edits during Summer 2024. The exceptional contribution of Claire Wagner to create both the online and PDF versions of this book must also be acknowledged. Generous support for this project from the Wheaton College Data Science Fund (Dr. Ying Li) and Summer Research Program is gratefully acknowledged.

As this work was completed at the time of my transition to professor emeritus, my hope and prayer is that this book will prove to be useful to the next generation of faculty and students.

P. Isihara, Wheaton College, Wheaton IL, July 2024

1.1 JMM 2023

The 2023 Joint Mathematics Meetings was held in Boston, Massachusetts. A Professional Enrichment Program (PEP) titled "Introductory Python Jupyter Notebooks for College Math Teachers" brought together members of the authorship team of this classroom resource project. The JNB "Patterns" was included as a 'hands-on' activity on the second day of the PEP. Browsing the contents of this notebook will give you some idea of the utility of JNBs. The remainder of the book will hopefully convince you that they are a useful tool for college math teachers and students.

1.2 JNB LAB: Patterns

> **Note**
>
> This lab gives an overview of some of the features that can be included in a Jupyter Notebook (JNB). This can be reviewed quickly before proceeding to the next chapter.

In this lab, we use the theme of 'Patterns' to introduce a variety of functions that can be done using JNBs from a pre-college to REU level:

I. PATTERNS IN NATURE (Pre-college)

1.1 Displaying an Image

1.2 Displaying a YouTube Video

II. PATTERNS IN SOCIAL DATA (Exploratory Data Analysis)

2.1 Reading in CPS Data from the Chicago Data Portal

2.2 Streamlining the Data

2.3 Simplifying the Column Names

2.4 Making a Scatterplot with OLS Regression line

III. PATTERNS IN MATHEMATICS (Undergrad math)

3.1 Tracing a Parametric Curve

3.2 Plotting Quadric Surfaces and Level Curves

IV.APPLYING MATHEMATICAL PATTERNS TO NATURE AND SOCIETY (REU)

4.1 Identifying Streamlines of Polluted and Freshwater Flow

1.2.1 Patterns in Nature

Patterns in nature are a beautiful and interesting phenomenon

Displaying an Image

a) Here is a NASA Hubble photograph of an exploding supernova. (The image files are available here).

```
from IPython.display import Image
Image(filename='supernova.png',width=100,height=100)
```

Exercise 1a

Display another beautiful natural pattern shown in the file 'MtFuji.png'.

Displaying a YouTube Video

Note

Please visit the web version of this section to engage with this interactive resource.

Patterns can arise arise in natural dynamical systems. For example, here is a scene of waves shown in the video "Ocean Waves." YouTube, uploaded by BusyBoy Productions , 14 February 2020, https://www.youtube.com/watch?v=J7pBztjUqUc&t=14s. Permissions: YouTube Terms of Service

```
from IPython.display import YouTubeVideo
YouTubeVideo('J7pBztjUqUc',width=100,height=100)
```

Exercise 1b

Display another video of a dynamic pattern from nature shown in the video

"Waterfall Clip." YouTube, uploaded by Bradley Erickson , 21 June 2016, https://www.youtube.com/watch?v=oYEtLQ3lEH0&t=5s. Permissions: YouTube Terms of Service

1.2.2 Patterns in Societal Data

Individual people, groups, and societies as a special part of the natural world give rise to patterns which can be analyzed using data. Here is an example using schools in zip code 60623. See the section *OLS Linear Regression*

Reading in CPS Data from the Chicago Data Portal

We use the Pandas (Python Data Analysis) library (abbreviated as pd) to read in the data. The '.head(2)' command displays the first two rows. We can also list the columns using the '.columns' command.

```
raw_CPS_data=  pd.read_json('https://data.cityofchicago.org/resource/kh4r-387c.json?
 ↪$limit=100000')
raw_CPS_data.head(2)
```

```
   school_id  legacy_unit_id  finance_id short_name  \
0     609966            3750       23531    HAMMOND
1     400069            4150       67081    POLARIS

                            long_name primary_category  is_high_school  \
0  Charles G Hammond Elementary School               ES           False
1              Polaris Charter Academy               ES           False

   is_middle_school  is_elementary_school  is_pre_school  ...  \
0              True                  True           True  ...
1              True                  True          False  ...

  fifth_contact_title fifth_contact_name seventh_contact_title  \
0                 NaN                NaN                   NaN
1                 NaN                NaN                   NaN

  seventh_contact_name refugee_services visual_impairments  \
0                  NaN              NaN                NaN
1                  NaN              NaN                NaN

  freshman_start_end_time sixth_contact_title  sixth_contact_name  \
0                     NaN                NaN                 NaN
1                     NaN                NaN                 NaN

   hard_of_hearing
0              NaN
1              NaN

[2 rows x 92 columns]
```

```
raw_CPS_data.columns
```

```
Index(['school_id', 'legacy_unit_id', 'finance_id', 'short_name', 'long_name',
       'primary_category', 'is_high_school', 'is_middle_school',
       'is_elementary_school', 'is_pre_school', 'summary',
       'administrator_title', 'administrator', 'secondary_contact_title',
       'secondary_contact', 'address', 'city', 'state', 'zip', 'phone', 'fax',
       'cps_school_profile', 'website', 'facebook', 'attendance_boundaries',
       'grades_offered_all', 'grades_offered', 'student_count_total',
       'student_count_low_income', 'student_count_special_ed',
       'student_count_english_learners', 'student_count_black',
       'student_count_hispanic', 'student_count_white', 'student_count_asian',
       'student_count_native_american', 'student_count_other_ethnicity',
       'student_count_asian_pacific', 'student_count_multi',
       'student_count_hawaiian_pacific', 'student_count_ethnicity_not',
       'statistics_description', 'demographic_description', 'dress_code',
       'prek_school_day', 'kindergarten_school_day', 'school_hours',
       'after_school_hours', 'earliest_drop_off_time', 'classroom_languages',
       'bilingual_services', 'title_1_eligible', 'preschool_inclusive',
       'preschool_instructional', 'transportation_bus', 'transportation_el',
       'school_latitude', 'school_longitude', 'overall_rating',
       'rating_status', 'rating_statement', 'classification_description',
       'school_year', 'third_contact_title', 'third_contact_name', 'network',
       'is_gocps_participant', 'is_gocps_prek', 'is_gocps_elementary',
       'is_gocps_high_school', 'open_for_enrollment_date', 'twitter',
       'youtube', 'pinterest', 'college_enrollment_rate_school',
       'college_enrollment_rate_mean', 'graduation_rate_school',
```

(continues on next page)

(continued from previous page)

```
       'graduation_rate_mean', 'significantly_modified',
       'transportation_metra', 'fourth_contact_title', 'fourth_contact_name',
       'fifth_contact_title', 'fifth_contact_name', 'seventh_contact_title',
       'seventh_contact_name', 'refugee_services', 'visual_impairments',
       'freshman_start_end_time', 'sixth_contact_title', 'sixth_contact_name',
       'hard_of_hearing'],
      dtype='object')
```

Streamlining the Data

In the next cell, we first create a dataframe df1 which includes just four columns: ['address','student_count_total','student_count_black','student_count_hispanic','student_count_white','zip'] and then filter to zipcode 60623. We then drop rows with missing data and give the total number of schools, as well as the largest and the smallest schools.

```
Total number of CPS schools considered in 60623 is 35
Largest student_count_total =  1072
Smallest student_count_total =  96
```

```
                 address  student_count_total  student_count_black  \
0         2819 W 21ST PL                  342                   33
1  2345 S CHRISTIANA AVE                  559                   66

   student_count_hispanic    zip
0                     304  60623
1                     484  60623
```

Simplifying the Column Names

We will abbreviate the column names as ["address","total","black","hispanic"]

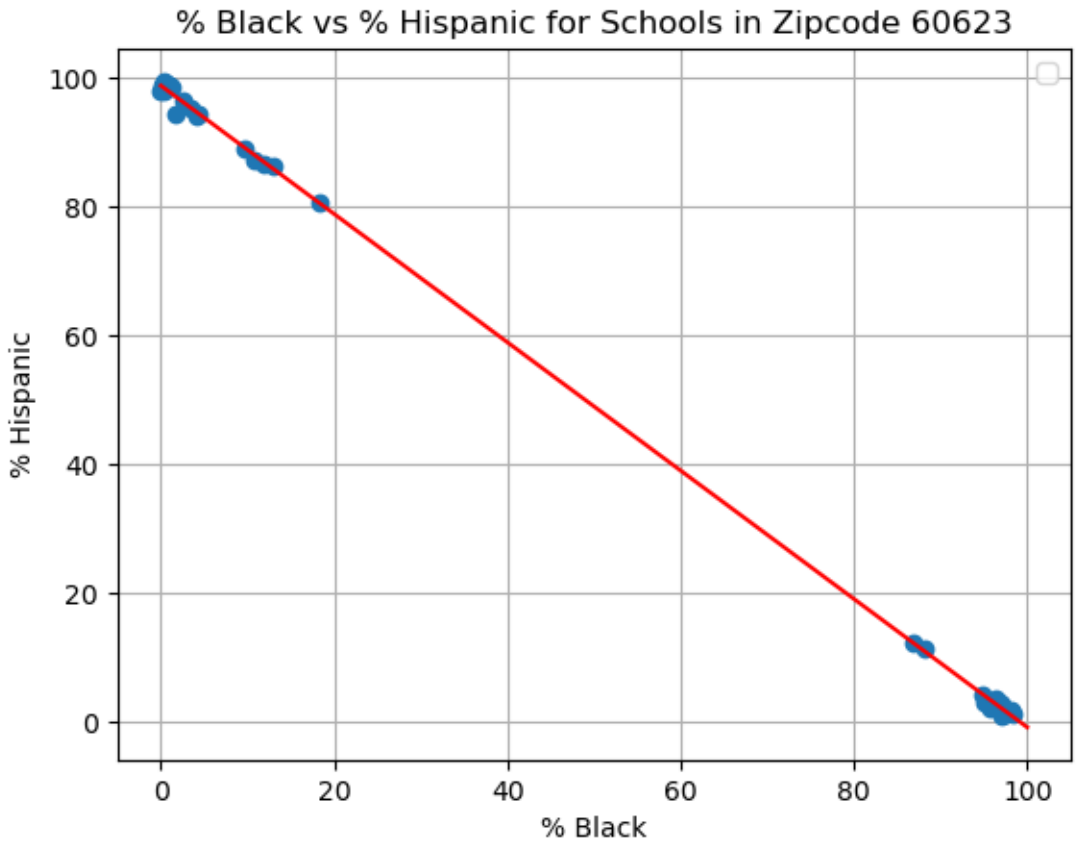
and then add 2 new columns '%black', '%hispanic'.

```
                 address  total  black  hispanic    zip  %black  %hispanic
0         2819 W 21ST PL    342     33       304  60623     9.6       88.9
1  2345 S CHRISTIANA AVE    559     66       484  60623    11.8       86.6
```

Making a Scatterplot with OLS Regression Line

The plot indicates %black (x-axis) vs. %hispanic (y-axis) and includes the OLS regression line on the plot

```
Intercept is  [98.71284906]
Slope is  [[-0.99515441]]
R^2 for OLS is  0.9996943205528053
```



This scatterplot shows that the CPS schools considered in 60623 are either predominantly hispanic or predominantly black.

Exercise 2

Modify the steps above to create a dataframe df2 and a scatterplot of % Black (x-axis) vs. % "graduation_rate_school' (y-axis) in zip 60623.

1.2.3 Patterns in Mathematics

Interesting and beautiful patterns arise throughout mathematics.

Tracing Parametric Curves

Various patterns can be created using sine, cosine, and exponential functions to define 2D parametric curves. (See the section [parametric equations](./Undergrad/Calculus/6 ParametricEquations.ipynb).)

The parametric equations

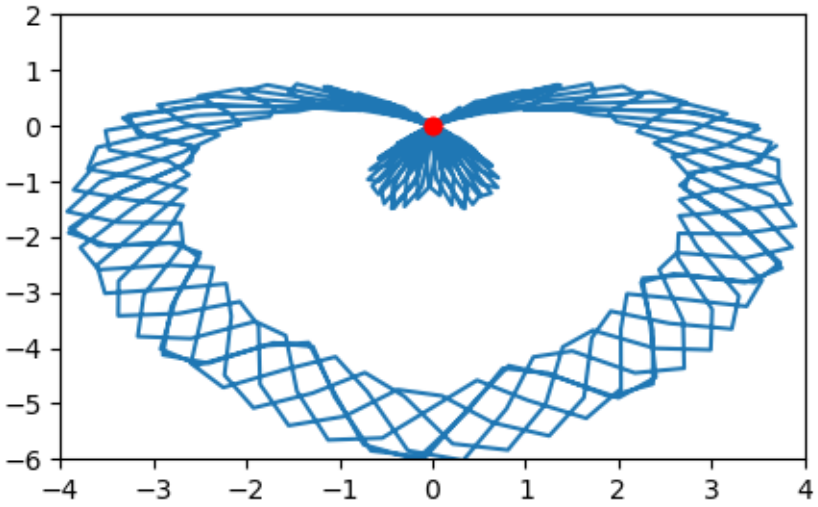
$x(t) = \cos(t)[3.5 - 1.5 \mid \cos t \mid \sqrt{1.3+ \mid \sin t \mid} + \cos(2t) - 3\sin(t) + .7\cos(12.2t)]$

$y(t) = \sin(t)[3.5 - 1.5 \mid \cos t \mid \sqrt{1.3+ \mid \sin t \mid} + \cos(2t) - 3\sin(t) + .7\cos(12.2t)]$

create a heart-shaped design.

Note

Please visit the web version of this section to engage with this interactive resource.



Exercise 3a

What familiar object resembles the parametric curve traced out by the equations

$x(t) = \sin(t)[e^{\cos t} - 2\cos(4t) - \sin^5(t/12)]$,

$y(t) = \cos(t)[e^{\cos t} - 2\cos(4t) - \sin^5(t/12)]$

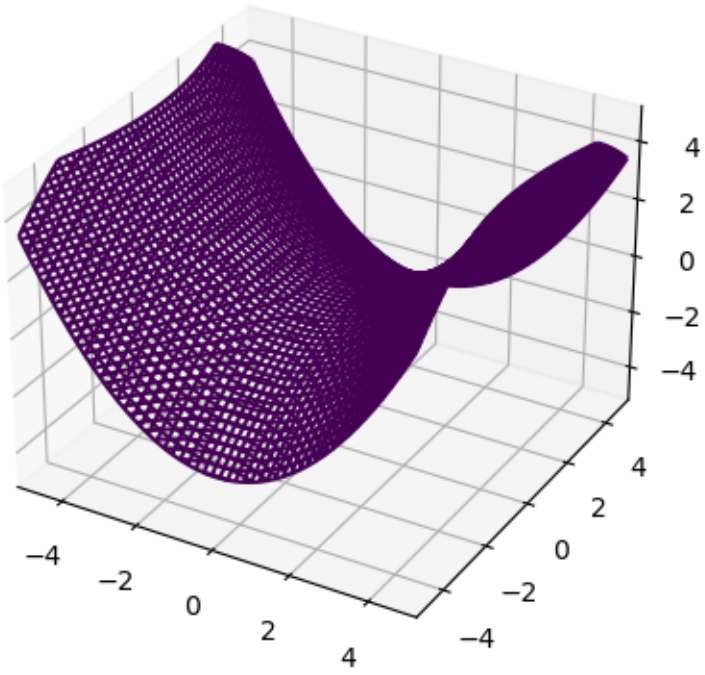
Plotting Quadric Surfaces and Level Curves

Quadric surfaces are another source of interesting patterns and can be constructed and analyzed using level curves. We can use the following function to plot quadric surfaces. The input variable fn is an expression in three variables (x,y,z). The function plot_implicit() plots the implicit relation fn=0 by plotting level curves parallel to the xy plane, xz plane, and yz plane.

For example, let us plot the elliptic paraboloid

$$z = (\frac{x}{2})^2 + (\frac{y}{3})^2 \tag{1.1}$$

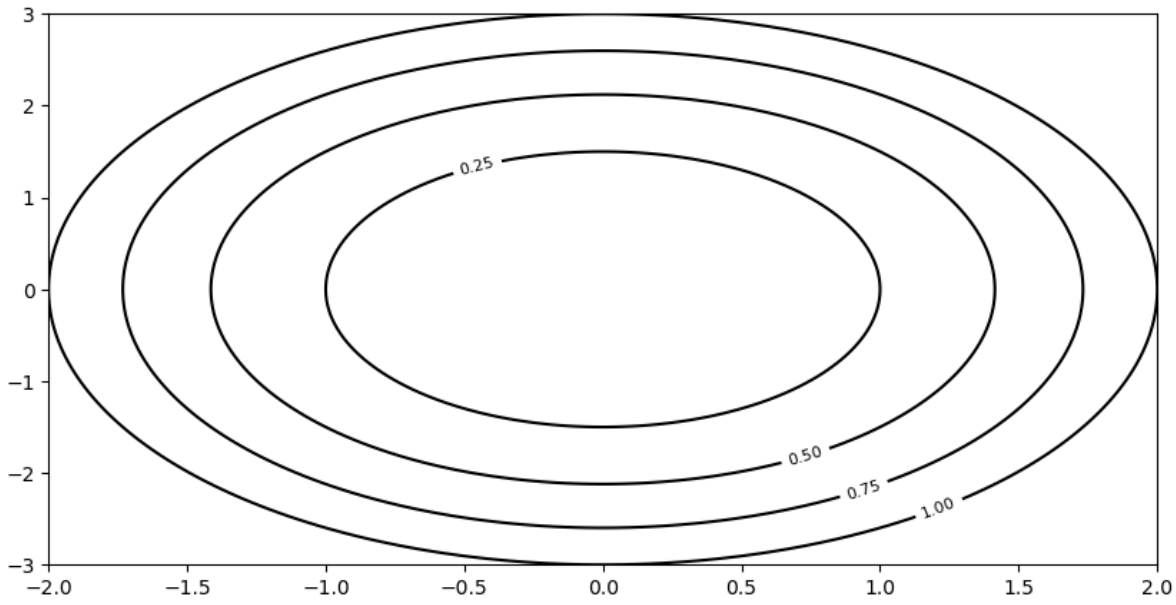
```
def hyperbolic_paraboloid(x,y,z):
    return (x/2)**2-(y/3)**2-z

plot_implicit(hyperbolic_paraboloid)
```



We can sketch the level curves of

$$z = (\frac{x}{2})^2 + (\frac{y}{3})^2$$

as follows:



Exercise 3b

Make a sketch of the hyperboloid of one sheet defined by

$(\frac{z}{4})^2 = (\frac{x}{2})^2 + (\frac{y}{3})^2 - 1.$

Then plot the level curves of the function

$z = 4\sqrt{(\frac{x}{2})^2 + (\frac{y}{3})^2 - 1}.$

1.2.4 Connecting Patterns in Mathematics to Patterns in Nature and Society

In complex variables, we study functions of a complex variable $f(z) = f(x + iy) = u(x, y) + iv(x, y)$ which maps a complex variable $z = x + iy$ represented by a point (x, y) in the complex plane to another complex variable $u + iv$ modeled as a point (u, v) in another complex plane.

Such functions have wide-ranging applications including applications to analysis of 2-dimensional fluid flow.

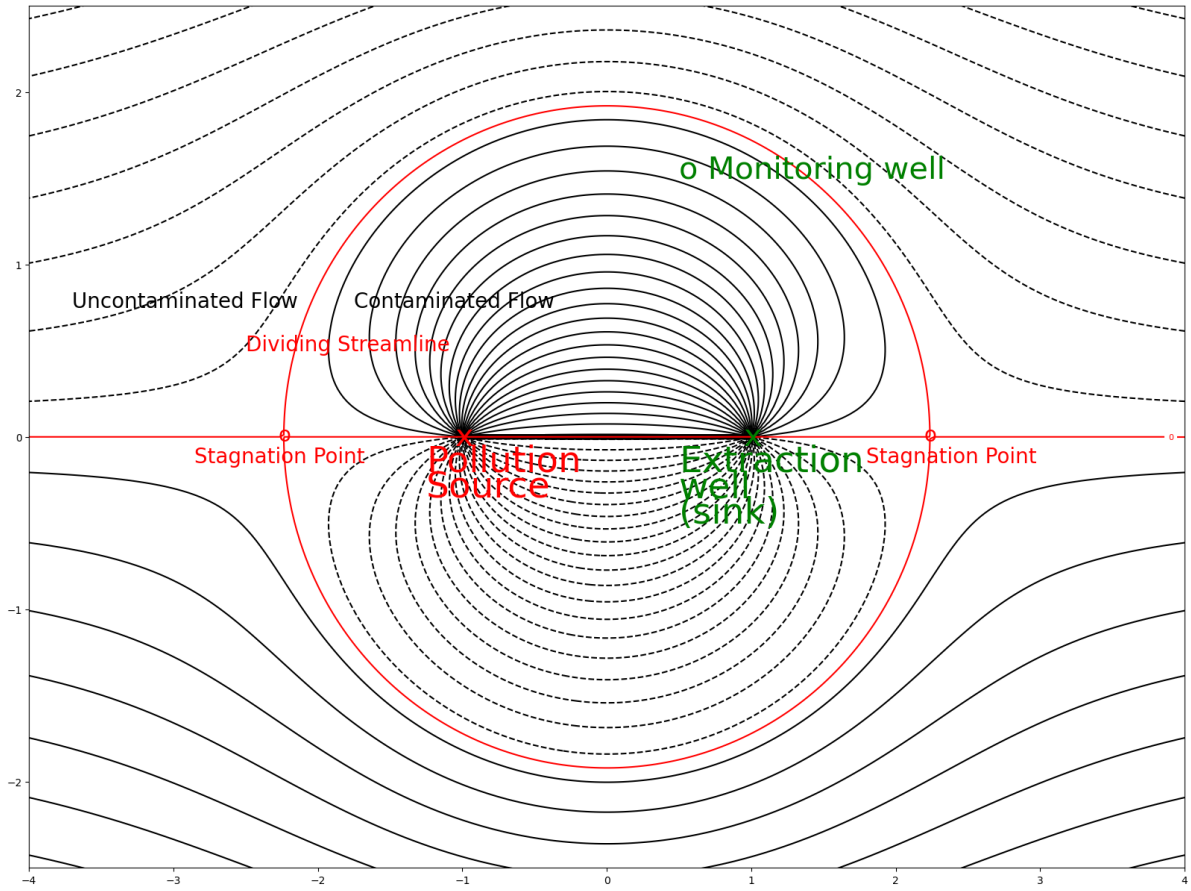
The following function $\Omega(z)$ called a "complex potential" can be used to model a scenario where there is a source of pollution located at (-1,0) as well an extraction well or sink of equal strength located at (1,0).

$$\Omega(z) = -z - 2\ln(z + 1) + 2\ln(z - 1)$$

The imaginary part of Ω is called the stream function Ψ. The level curves of Ψ are called streamlines, and model the path along which the fluid particles flow.

Identifying Streamlines of Polluted and Freshwater Flow

We now make a plot of the streamlines. A Rankine Oval separates the freshwater streamlines from the polluted streamlines.



Note that the extraction well is sufficiently strong to capture all the pollution streamlines. The boundary between the polluted and freshwater streamlines is called a "Rankine Oval."

Exercise 4

Read the paragraph "Model Output" in Section 4 "Contaminant Extraction Modeling" of the The COMPLEX VARIABLES IN GROUNDWATER MODELING chapter. Then explain the difference between an Ineffective, Regular, and Inefficient system.

1.3 JNB Lab Solutions

1.3.1 Patterns in Nature

Solution to Exercise 1a

Solution to Exercise 1b

"Waterfall Clip." YouTube, uploaded by Bradley Erickson , 21 June 2016, https://www.youtube.com/watch?v=oYEtLQ3lEH0&t=5s. Permissions: YouTube Terms of Service

1.3.2 Patterns in Societal Data

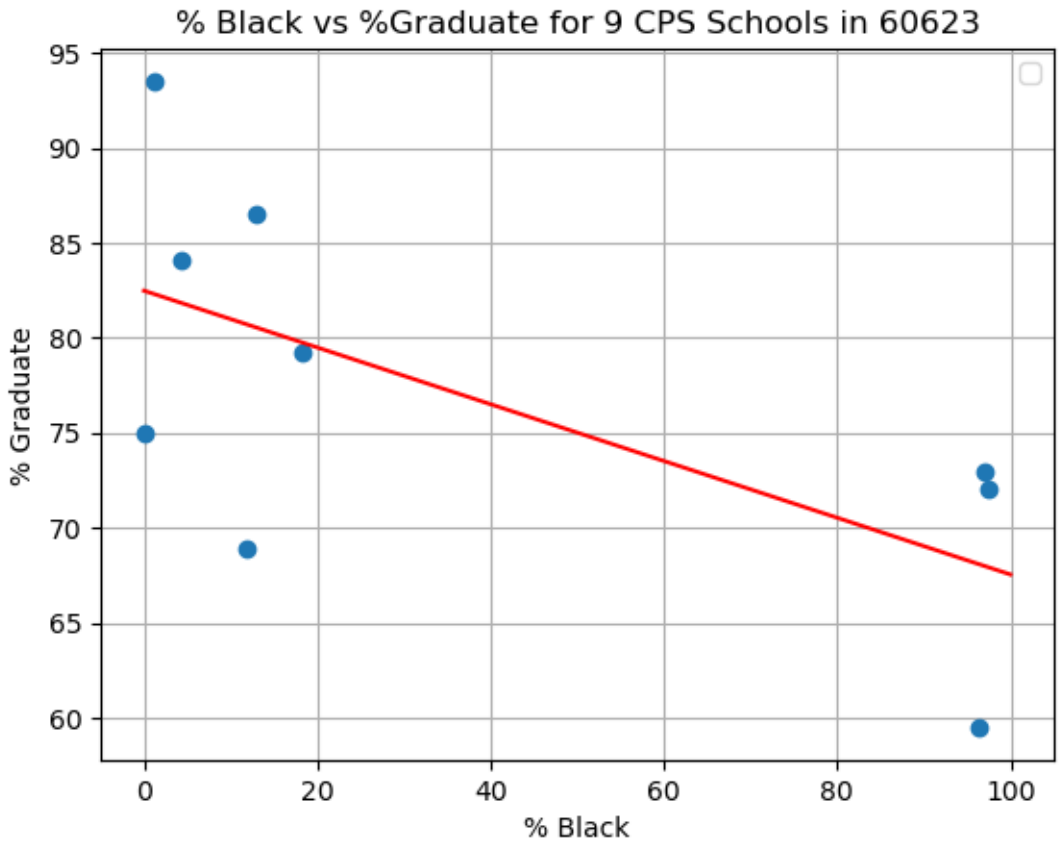
Solution to Exercise 2

```
Total number of CPS schools considered in 60623 is 9
Largest student_count_total =  559
Smallest student_count_total =  100
```

```
                address  student_count_total  student_count_black  \
0  2345 S CHRISTIANA AVE                  559                   66
1     3120 S KOSTNER AVE                  319                   41

   graduation_rate_school    zip
0                    68.9  60623
1                    86.5  60623
```

```
                address  total  black  graduate    zip  %black
0  2345 S CHRISTIANA AVE    559     66      68.9  60623    11.8
1     3120 S KOSTNER AVE    319     41      86.5  60623    12.9
```

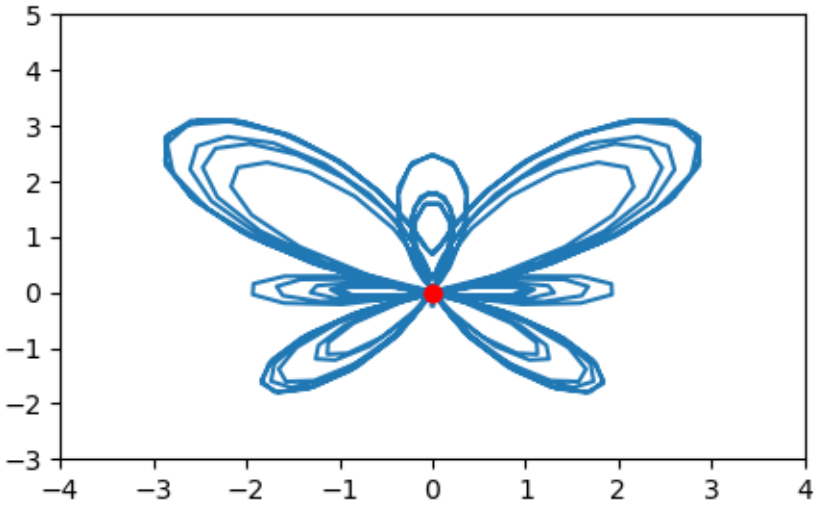
```
Intercept is  [82.47649769]
Slope is  [[-0.14939133]]
R^2 for OLS is  0.429488206625943
```



This shows graduates rates tend not to be as strong in the predominantly black schools in 60623.

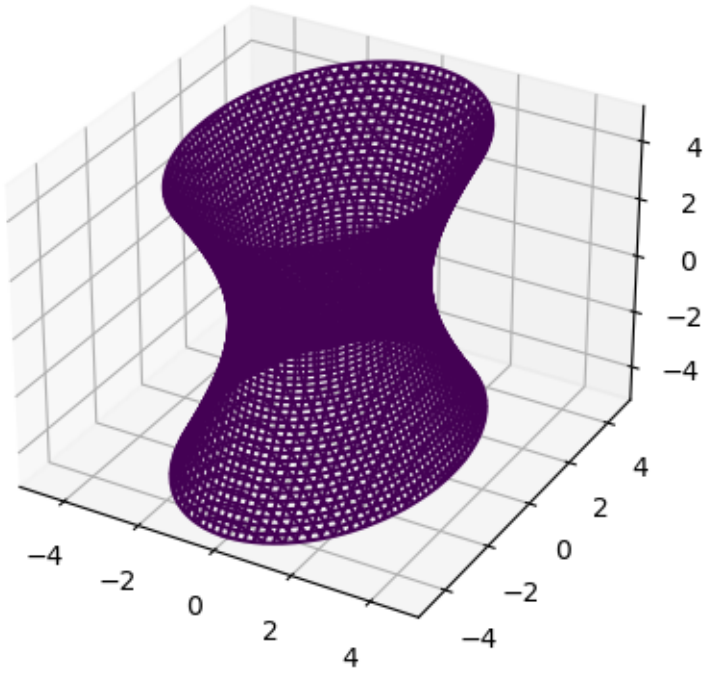
1.3.3 Patterns in Mathematics

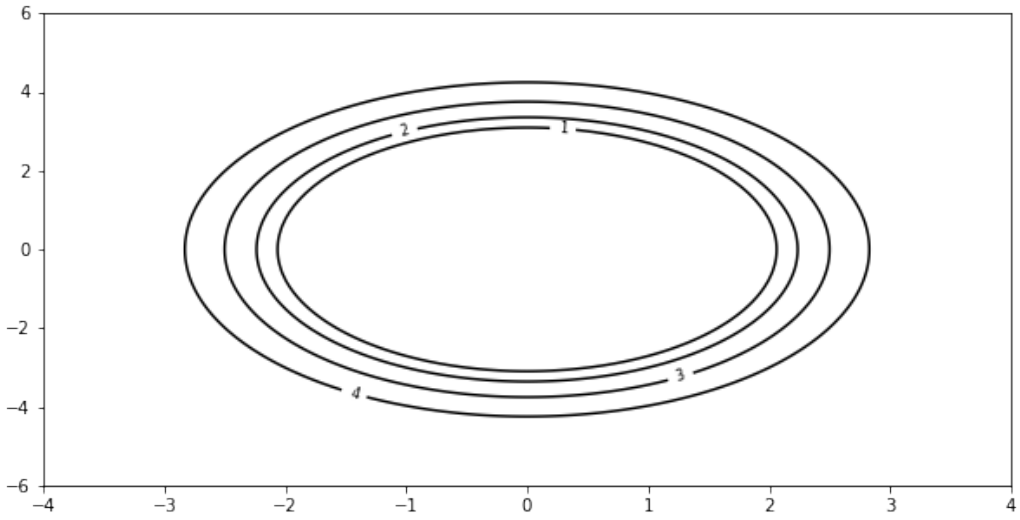
Solution to Exercise 3a



Solution to Exercise 3b.

```
def hyperboloid_1sheet(x,y,z):
    return (x/2)**2+(y/3)**2-(z/4)**2-1

plot_implicit(hyperboloid_1sheet)
```





1.3.4 Connecting Patterns in Mathematics to Patterns in Nature and Society

Solution to Exercise 4.

An ineffective system is one in which the two extraction wells (one mid-plume and one midstream) are not strong enough to capture all the pollution detected by the monitoring wells.

A regular system is one where the two extractions working together are sufficient to capture all the pollution.

An inefficient system is one where the mid-plume extraction well could be turned off, and the downstream extraction well by itself is sufficiently strong to capture all the pollution.

CHAPTER 2

PRE-COLLEGE - Wheaton College Team

Note

The material in this chapter is a partial compilation of labs that were used in after-school programs at Cornerstone Academy and Celestial Ministries in Chicago, a math for the benefit of society general education course at Wheaton College (IL), and a research paper on the Chicago Hardship Index. The idea for these labs originated in Summer 2020 by a Wheaton College (IL) team comprised of Dr. Paul Isihara (professor), Hope Wood (Emerald South Scholars staff) and Demetrius Crawford (Emerald South Scholars student) with subsequent support by Dr. Peter Jantsch (assisting professor) along with Christian Cameron, Sean Ferguson, Eric Fraatz, Lucy Henneker, Andy Holmberg, Natan Jara, Ethan Kile, Gracie Johnson, Alex McKeaney, Olivia Pearsall, Andy Peterson, Christopher Plimpton, Daniel Rendall, Kiki Rogers, Nathan Varberg, Willow Vander Kooi, and Kirsten Wikner (Wheaton students).

2.1 Introduction

This chapter is a compilation of pre-college and general education level material intended to introduce students to JNBs and basic principles of programming. The sections include topics such as:

- Blackboard Problems: these examples are designed as an overview of basic mathematical operations for early learners;
- Beginning Python Programming: this section details how to use JNBs and familiarizes students with basic programming principles and terminology;
- Glimpse of Chicago: this material was first developed as a lab/lesson for middle and high school students in after school programs in Chicago to learn the basics of how to use JNBs;
- Arts in Steam: similar to the last section, this section seeks to provide an overview of the functionality of JNBs by combining their use in stem with various artistic interests (music, image editing, graphical design) to attempt to help interest students who may not otherwise have much interest in programming; and
- JNB LAB- After-School Program Demo: this serves as a comprehensive lab that students can work through after having completed the previous material that introduced them to JNBs.

2.2 Elementary Blackboard Problems

Note

One of the two main motivations for this book is to support the rebuilding of the education system in Tigray, Ethiopia, which suffered a horrific 20 year civil war from November 2020 - November 2022. At the time of writing of this book, over 2,000 schools were closed and 2 million children out of school, most of them for over 2 years. Student to classroom ratios exceeded 400:1! As schools buildings were being repaired, the need for blackboards was deemed even more basic than the need for students to have paper and pencils. This "low-tech" section addresses the question: "If you had just a blackboard to teach elementary level math, what type of problems would you put on the board?"

2.2.1 Arithmetic

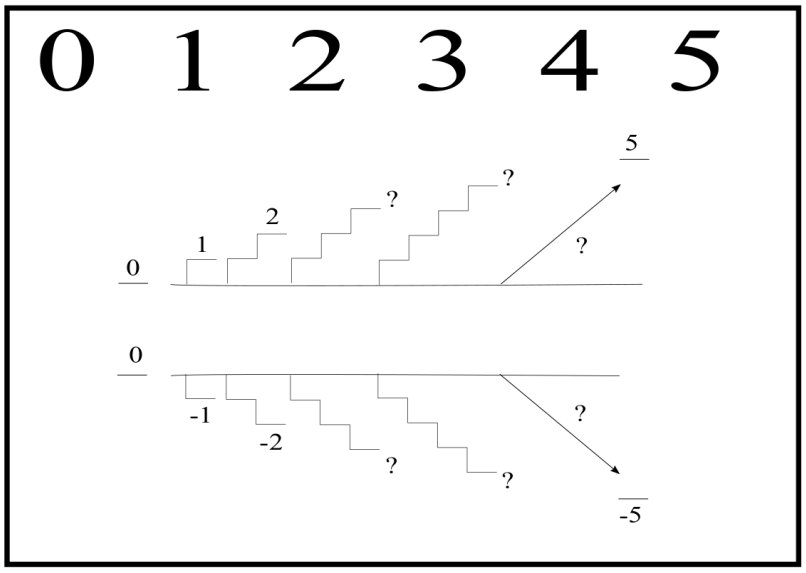
ሕሳብ

Positive and Negative Numbers

1. ኣወንታውን ኣሉታውን ቍጽርታት

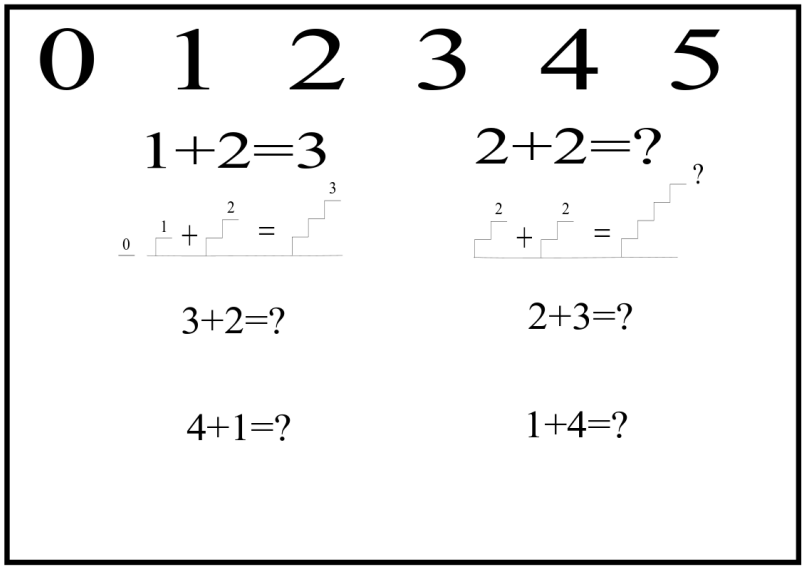
Note

The German mathematician Leopold Kronecker (1823-1891) said: "God made the integers; all else is the work of man." This set of blackboard puzzles is designed to help young students understand the four basic arithmetic operations (addition, subtraction, multiplication and division) staying within the set of integers {…-3,-2,-1,0,1,2,3,…}



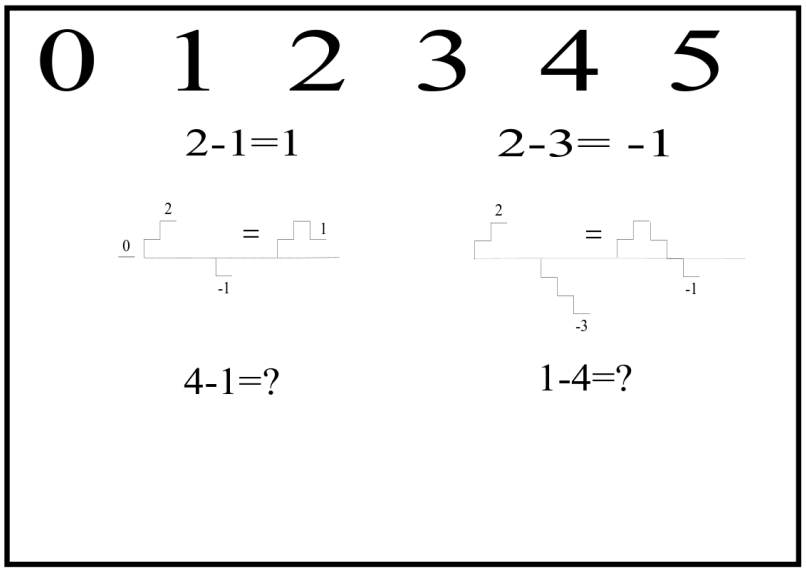
Addition

2. መወሰኽታ



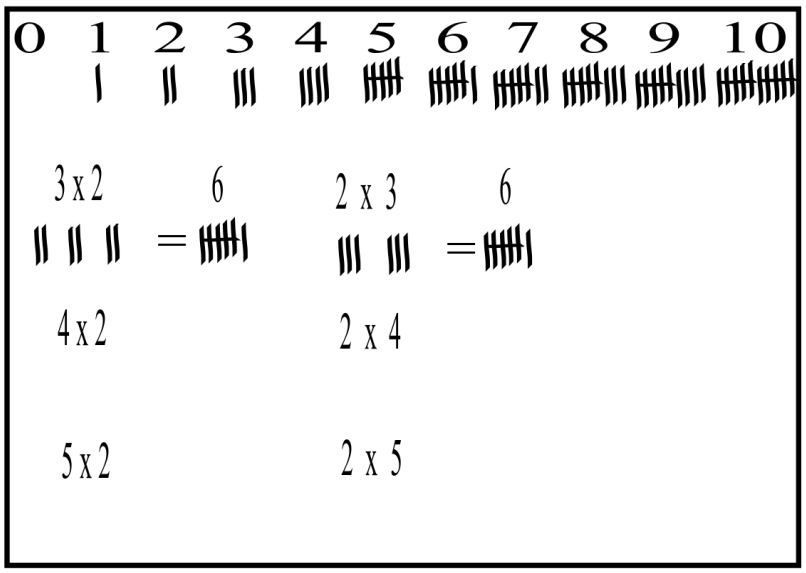
Subtraction

3. ንኡስ ኣርእስቲ



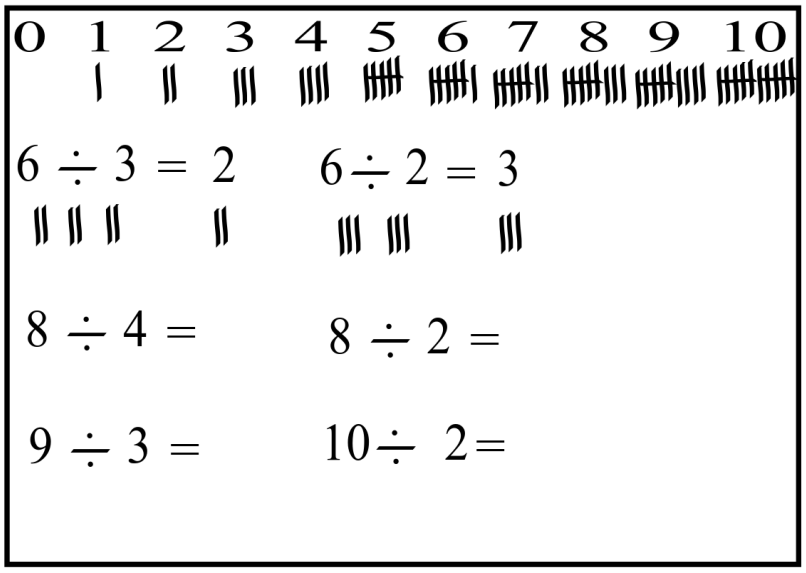
Multiplication

4. ምብዝሑ



Division

5. ምክፍፋል

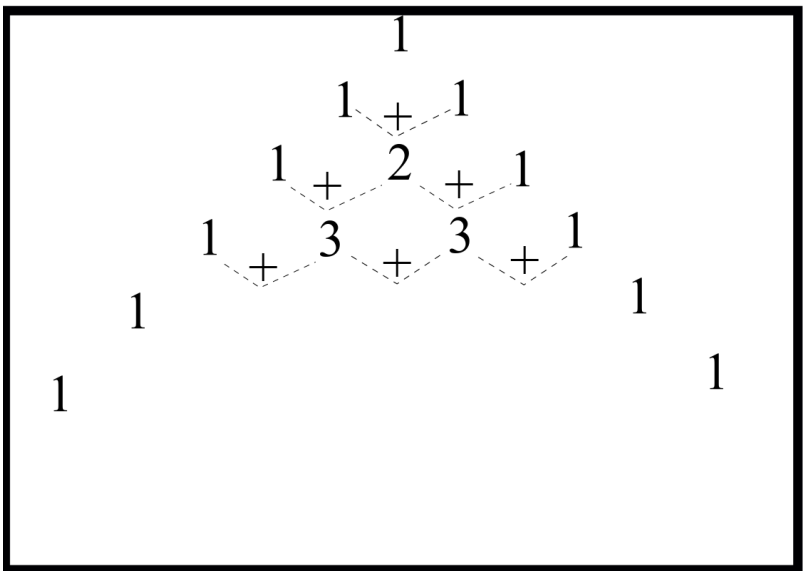


2.2.2 Pascal's Triangle

ስሉስ ኮርናዕ ፓስካል

Note

Pascal's triangle is used to determine the coefficients in a binomial expansion (x+y)^n.
For example, if n=1, the coefficients are 1,1: (x+y)^1= 1 x + 1 y. If n=2, the coefficients are 1,2,1: (x+y)^2= 1 x^2 + 2 xy + 1 y^2. Elementary students may construct Pascal's triangle once they see the pattern.

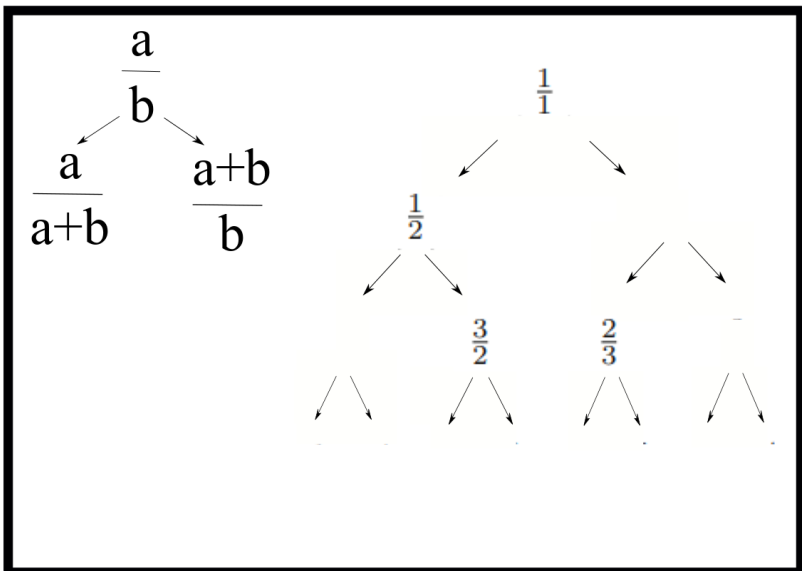


2.2.3 Fractions

ቕርጽራጽ

Note

The tree diagram below can be used to enumerate all the positive fractions. Each fraction appears exactly once in the tree. Construction of the tree requires only elementary addition of positive numbers.

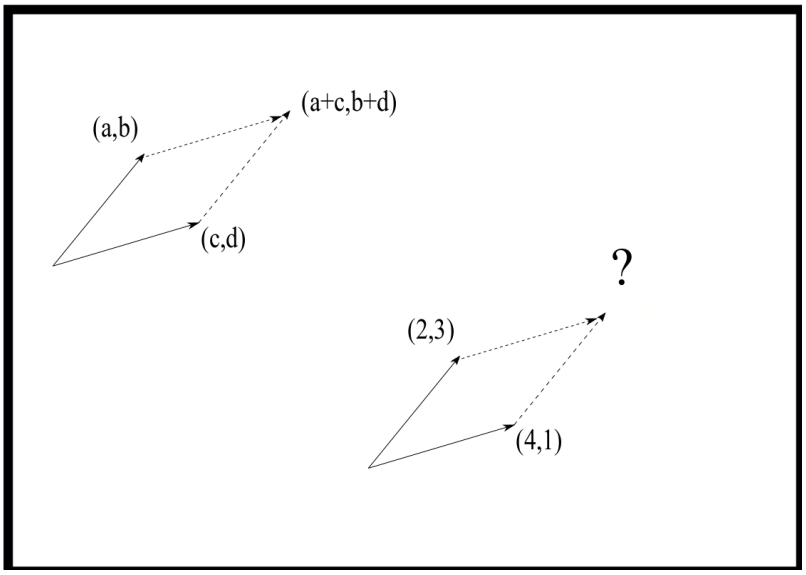


2.2.4 Vectors

ወተሃደራት

Note

Elementary addition is also basic to the rule for addition of 2 dimensional (plane) vectors: simply add together the respective 'x' and 'y' components of the two vectors. This process can be represented in an (x,y) coordinate plane. Arithmetic and geometry are combined in the parallelogram representing the sum of two vectors.

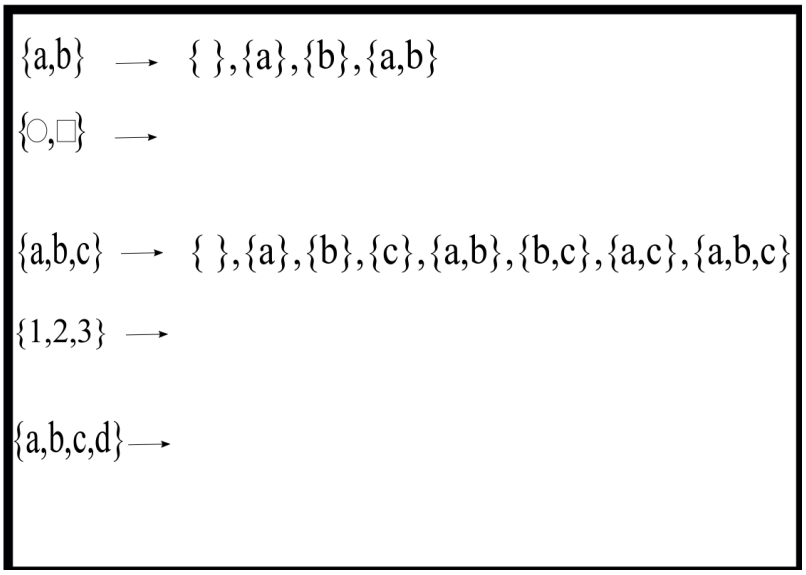


2.2.5 Subsets

ንኡስ ሰሌዳታት

Note

Mathematics may be described as the science of patterns. In this puzzle, the pattern for subsets of a set is introduced for a 2 and a 3 element set. The student must then generalize the pattern to a 4 element set.

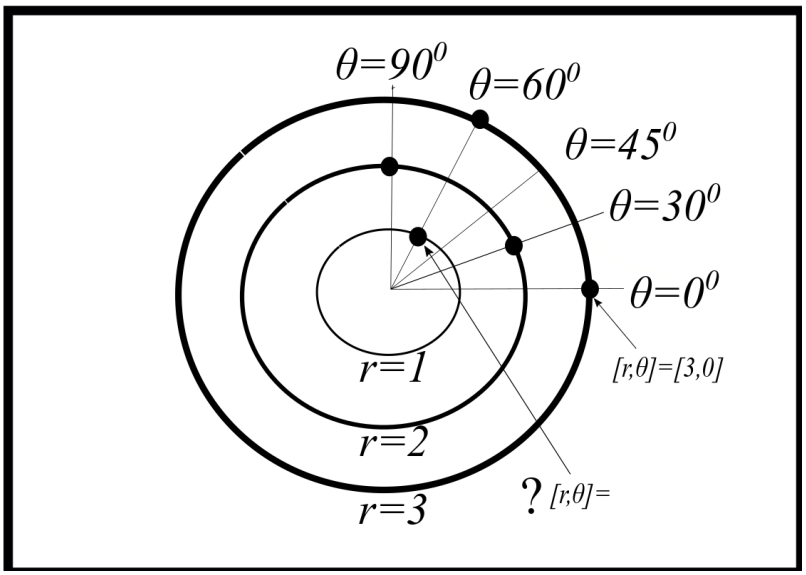


2.2.6 Polar Coordinates

ኣወሃህባታት ዋልታ

Note

Polar coordinates offer a different way to describe the location of a point in a plane. The concepts of distance and angle are intrinsic to polar coordinates.

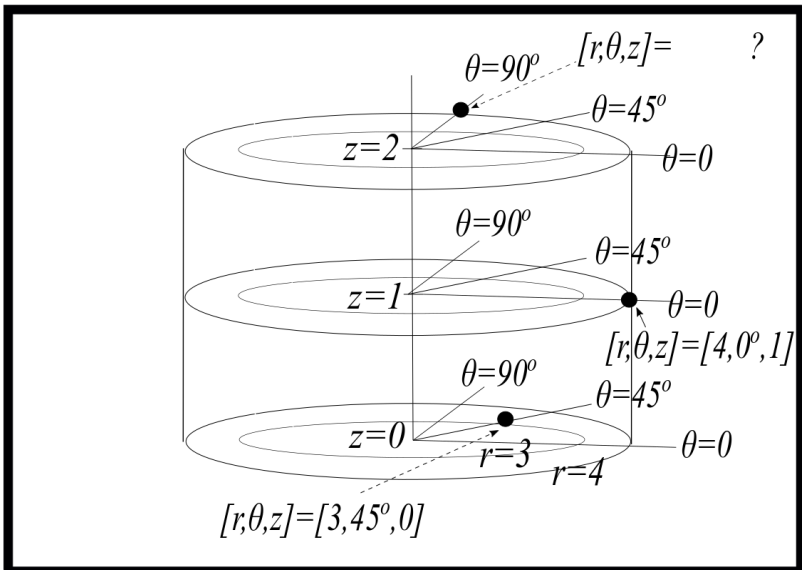


2.2.7 Cylindrical Coordinates

ሲሊንደራዊ ኣውሃህባታት

Note

Cylindrical coordinates extend the idea of polar coordinates, and introduce the idea of locating points in a 3-dimensional space.



2.2.8 Symmetry

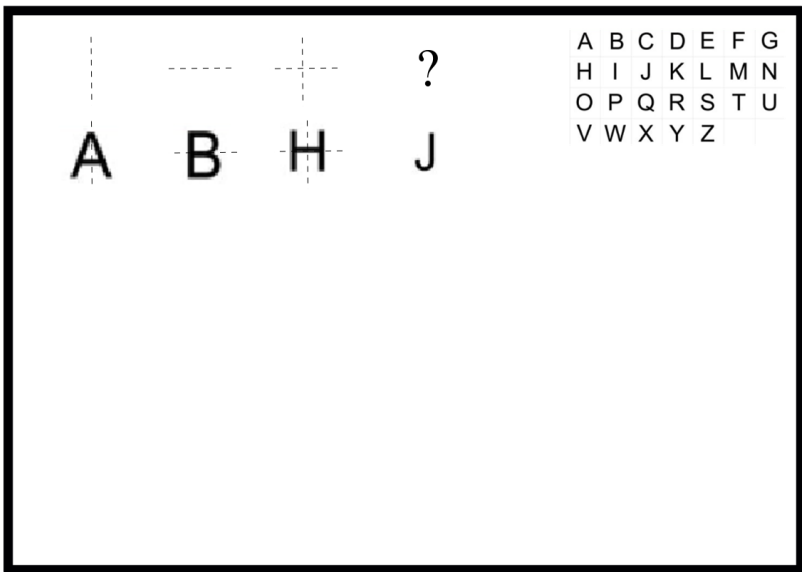
ምምስሳል

Note

A mathematical symmetry is an operation which leaves the original object unchanged. Letters in the English alphabet can be classified according to vertical symmetry (reflection across a vertical line), horizontal symmetry (reflection across a horizontal line), both, or neither.

English Alphabet

ፊደላት እንግሊዝኛ

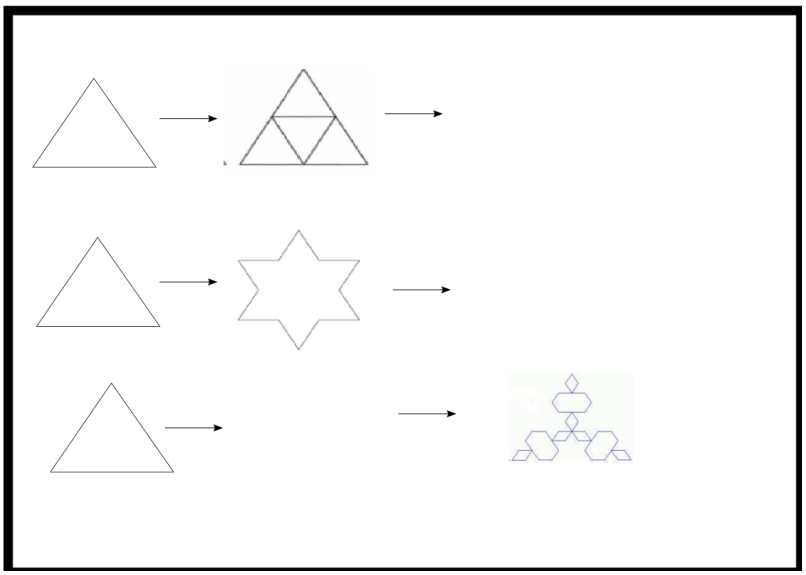


2.2.9 Fractals

ፍራክታል

Note

Fractal geometry creates many interesting shapes. One type of fractal pattern is obtained by an iterative process. For example, an equilateral triangle may be subdivided into 4 smaller equilateral triangles. In turn, each smaller equilateral triangle can be subdivided into 4 even smaller equilateral triangles, and so on. The goal of the puzzle is to figure out the iterative process for each row, and apply it.

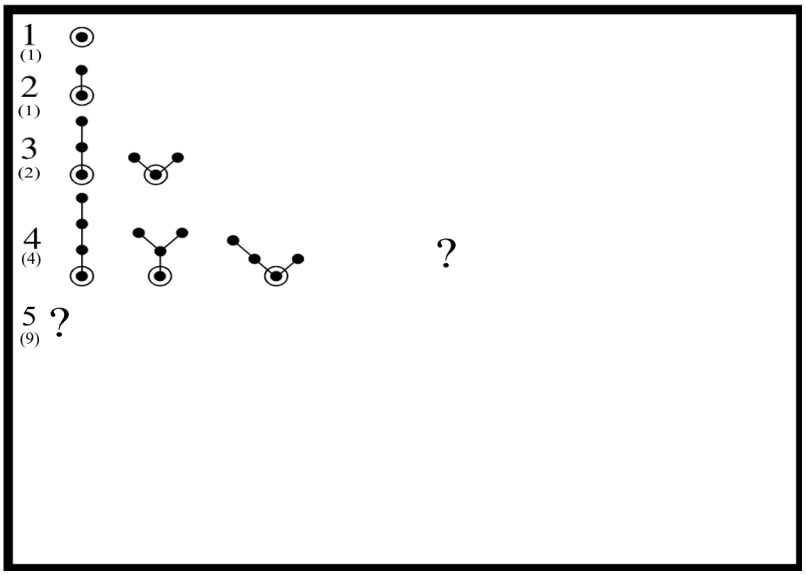


2.2.10 Trees

ክንደይ እተፈላለየ ኣእዋም እዩ ዘሎ ?

Note

The goal of this puzzle is to enumerate all the different types of rooted trees with n nodes. The root of each tree is indicated by the circled node. A tree is a connected graph which has n nodes (vertices) and n-1 edges.

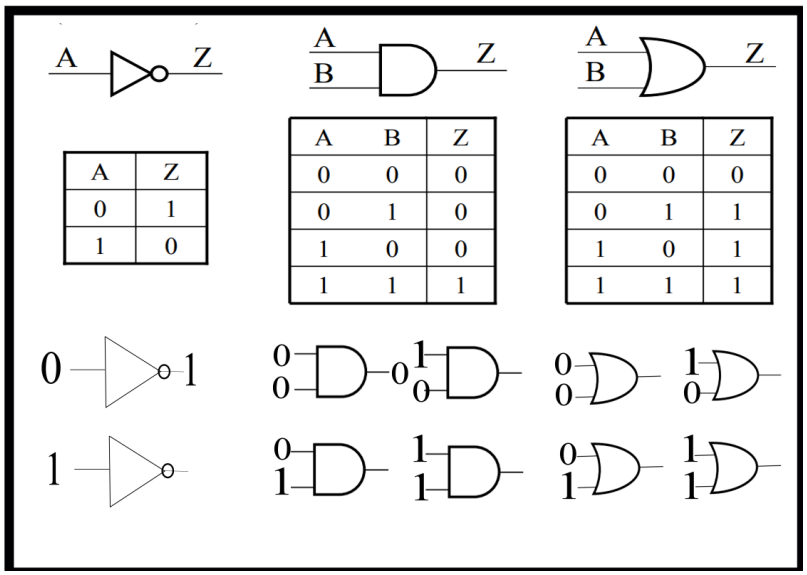


2.2.11 Circuits

ወረዳታት

Note

In a logic circuit, wires carry the values 0 (“low voltage”) or 1 (“high voltage”). Three basic gates through which wires may pass are a “NOT gate”, “AND gate”, and “OR gate”. Depending on the input value(s), the output value(s) of each type of gate are indicated by the tables below.



A	Z
0	1
1	0

A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

2.3 Beginning Python Programming

2.3.1 Getting Started with JNBs

Note

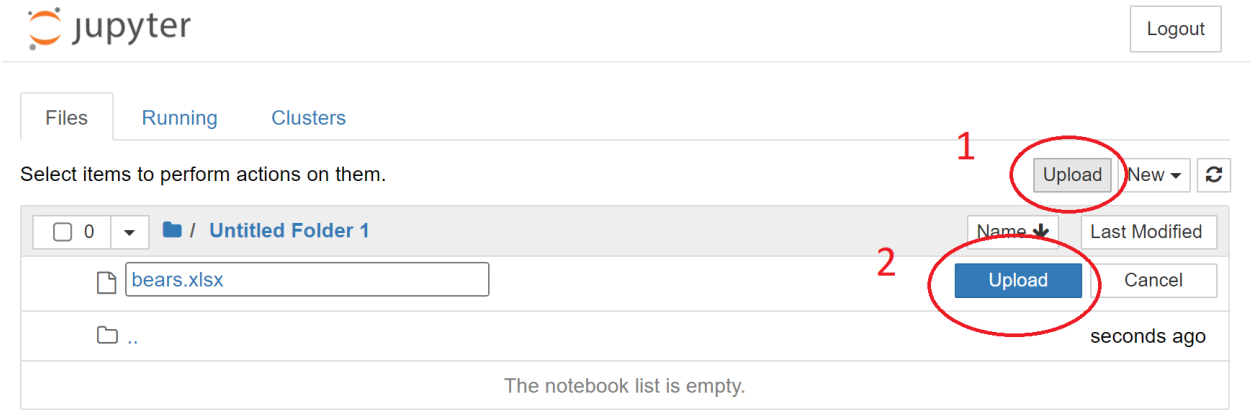
We begin with an outline of how you can install and run a Jupyter Notebook (JNB) on your computer.

1) Begin by downloading the free Anaconda distribution from www.anaconda.com
2) Register an account and open the application Jupyter Notebook using Anaconda
3) Next, download two files from https://drive.google.com/drive/folders/1cpRptbyjYZSENEtjNNEiV7tBrIz4mONI?usp=sharing

(i) the Jupyter Notebook “Getting Started with Jupyter Notebooks.ipynb”

(ii) the Excel file “bears.xlsx”

4) Upload the two files in step two into your JNB directory:



5) Open the JNB "Getting Started with Jupyter Notebooks" and follow the instructions.

2.3.2 Functions

Note

The Python program language is one of the most popular worldwide. It is used widely by data scientists, a profession that is in high demand and well paid if you are good at it. In this demo, we introduce the concept of a function.

1) Here we define a function called add(a,b) that computes the sum a+b. Click on the next cell and press the Run key at the top menu bar (or alternately, hit shift+enter simultaneously) to execute the next cell which creates our function. The # symbol followed by a comment is not needed to execute the code.

```
def add(a,b):    #make sure you include the :
    x = a+b    #make sure you indent
    return x   #return is a key word found at the last line of the code defining the␣
↪function
```

2) Here is an example of how we can use the function we just created to compute 5+3. (Always click on the cell and press Run or hit shift+enter to execute the cell).

```
add(5,3)
```

3) Now it's your turn. Define a function called subtract(a,b) which computes a-b. Then test that your function is working by showing subtract(5,3) is equal to 5-3=2.

```
#Define your function subtract(a,b) in this cell
```

```
#Test your function by computing subtract(5,3) in this cell. (Make sure you pressed␣
↪shift+enter in the previous cell.)
```

4) Now read over and then execute the next cell. Remember # is the start of a comment.

```
# Let's define a function multiply(a,b) and then test it on multiply(5,3)
def multiply(a,b):
    x=a*b
    return x
multiply(5,3)
```

5) Now it's your turn. Define a function divide(a,b) and test it on divide(5,3). Note that the symbol for divide is /

```
# Define your function divide(a,b) in this cell
```

```
# Test your function by computing divide(5,3) in this cell
```

6) CHALLENGE 1: Can you solve the following math problem by providing the correct input to the Python function? Study the code below average(a,b,c). Then given that a=2, b=5, figure out the correct value of c so that average(a,b,c)=10.

```
def average(a,b,c):
    ave = (a+b+c)/3
    return ave
```

```
a=2
b=5
c=20 #Here's a guess for c.  Edit it so you get the right answer.
average(a,b,c)
```

Congratulations if you defined c in such a way as to get 10.0.

Note

Your instructor may ask you to submit a file of your completed work.

You can create a PDF of your completed JNB by selecting File on the top menu bar, and then Print Preview. Right-click, and then print to PDF.

You can also download the JNB itself by selecting File, Download as, Notebook(.ipynb).

Try creating both file formats.

Congratulations! You have successfully completed this intro to functions!

2.3.3 numpy

Note

In the previous section, we created a few simple functions to get used to how we can create a 'special work order' to get the computer to do something for us.

Certain functions are used so often and widely that it really makes no sense for everyone to re-invent the wheel each time by defining a new function. Instead, there are 'libraries' of functions which can be 'imported' for use in our Jupyter Notebook (JNB). We can use any function in an imported library by knowing the name of the function and what inputs are required.

The first library we will use is the 'numerical Python' library called 'numpy' and abbreviated as 'np'.

(1) Click on the next cell and then press run (or shift+enter) to import the numpy library.

```
import numpy as np  #this is how we import the numpy library with abbreviated name np
```

The . extension

(2) We can access a function in the numpy library using

np.

followed by the name of the function.

For example, suppose we wish to create a list of numbers 0,10,20,30,40,50,60,70,80,90,100. There is a numpy library function arange(starting_number,gone_too_far_number,spacing_by) which works well for this task.

In our case, use starting_number=0, gone_too_far_number=110, spacing_by=10.

Remembering to put np. in front of the function, execute the next cell (press run or shift+enter) to see the result.

```
np.arange(0,110,10)
```

(3) Now you try it. Use numpy to make a list of all the numbers between 0 and 50.

(4) Inside of a library there might be sub-libraries, akin to a children's library within a main library. One of numpy's sub-libraries is called random.

SINGLE CHOICE POP QUIZ How do we access the random library within the numpy library?
a) np.random

If you answered a) np.random you are correct!

Inside the random library is a function called randint(numbers_in_a_hat).

The input value for numbers_in_a_hat is a positive integer like 10. This specifies that ten numbers starting with 0 (i.e. 0,1,2,…,9) will put into a hat.

The function randint(10) then tells the computer to pull out randomly one of the numbers in our hat.

Remembering to put np.random. in front of the function, try it by hitting shift+enter several times in succession on the cell below to simulate random draws from a hat (with replacement).

```
np.random.randint(10)
```

(5) Now it's your turn. Tell the computer to pick a random number from a hat which has the numbers 0,2,3,…,999. Do it 3 times to see if you get the same number each time.

```
#First Try
np.random.randint(1000)
```

```
#Second Try
np.random.randint(1000)
```

```
#Third Try
np.random.randint(1000)
```

Hope you enjoyed this quick look at the numpy library!

2.3.4 pandas

Note

Python's data analysis library is called pandas and usually abbreviated as pd. Pandas is one of the most important tools used by data scientists today.

1) Let's begin by importing the library in the same way that we imported the numpy library. (Remember to click on the next cell and then press run (or shift+enter) to execute the next cell.)

```
import pandas as pd
```

File upload

Get the file bears.xlsx from the folder https://drive.google.com/drive/folders/1cpRptbyjYZSENEtjNNEiV7tBrIz4mONI?usp=sharing

Then put the file in the same directory as this Jupyter notebook. (You can do this by choosing File on the top left of the Jupyter Notebook menu bar, and then selecting Open. Then hit the Upload button on the top right, find bears.xlsx, hit Open and then Upload.) Return to this Notebook when you have completed the upload of the datafile.

2) Now we can use the read_excel() function in the pandas library to read the data into what is called a dataframe. We will name our dataframe bear_info.

```
bear_info=pd.read_excel("bears.xlsx")  #read in the data
bear_info  #display the data
```

We can set the names of the columns using the following command.

```
bear_info.columns=["Type","Age","Size","Weight"]
bear_info
```

4) We can display just the first two rows of our dataframe using a command of the form df.head(2) where df is the name of our dataframe.

```
bear_info.head(2)
```

5) The column at the left is called the index. Note that the index values start with 0. We can locate information in the dataframe using a command of the form df.loc[index,column] where df is the name of our dataframe, index is the value of the index, and column is the name of the column inside of quotation marks. Here is an example.

```
bear_info.loc[1,"Weight"]
```

6) Use .loc() to get the age of a Grizzly bear.

```
#your answer to 6)
```

7) Use .loc to get the size of a Black bear.

```
#your answer to 7)
```

Exercises

Exercises

1) Create your own data file with some interesting info. Read in the data into a dataframe called df using pd.read_excel().
2) Display the first row of df.
3) Abbreviate the column names.
4) Display the first line of your dataframe with abbreviated column names.
5) Show how to use .loc() to get a particular entry in your dataframe.

2.3.5 matplotlib

Note

Matplotlib was created by John Hunter to help make graphs. Like numpy and pandas, matplotlib is a widely used Python library. Dr. Hunter used this library as a neurobiologist, studying epilepsy at the University of Chicago. Unfortunately, Dr. Hunter died of cancer at age 44.

1) The matplotlib library has a sub-library called pyplot (plt). We can access the functions in pyplot by executing the next cell.

```
import matplotlib.pyplot as plt
```

2) Here is an example of a simple plot.

```
# create a new figure
plt.figure()
# plot the points (0, 1) (1,0),(2,1),(3,0) and (4,1) with a 'o' marker and connect␣
 ↪them using 'o'
xvalues=[0,1,2,3,4]
yvalues=[1,0,1,0,1]
plt.plot(xvalues, yvalues, 'o-',color='blue')
```

3) Use pyplot to create a red letter M.

```
#Your answer to 3)
```

4) Let's make a simple bargraph using pyplot.

```
plt.figure()
xvals = [0,1,2]
heights=[2,4,6]
plt.bar(xvals, heights, width = 0.3,color='black')
```

5) Create a bar graph with 4 bars at x positions 1,2,3 and 4 and with heights 2,3,1,5.

```
# Your answer to 5)
```

6) One more example is a pie chart.

```
plt.figure(figsize=(5,5)) #you can adjust the figure size
activities=["work","play","eat","sleep"]
hours=[8,3,3,10]
plt.pie(hours,labels=activities,autopct='%1.1f%%')  #percentages are automatically␣
↪computed
```

7) Make a piechart which describes your usual daily activities and how long you spend on each.

```
#Your answer to 7)
```

2.3.6 for loops

Note

In this section, we will introduce a "for loop" as a tool to build more complicated user-defined functions. We will also make use of the numpy library function arange(). (It might be good to go back and review the previous sections.)

1) Let's start by importing the numpy library. (Don't forget to press shift+enter to execute each cell.)

```
import numpy as np
```

2) Let's create a list of all numbers between 1 and 10.

```
my_list=np.arange(1,11,1)
my_list
```

3) Python has technical names for the things we create. The built-in type() function will give us the technical name. Let's check the technical name for our list.

```
type(my_list)
```

4) Let's define a function which takes an input list and outputs the square of each number in the list. A for loop is used to go through one by one the numbers in a list.

```
def squared(my_list):  #remember the use of :
    for i in my_list:  #must indent lines in a function; for loops need a :
        print(i**2)       #must also indent lines in a for loop
    return print("Finished!")  #prints a message to the screen when completed
```

5) Let's see if this works.

```
squared(my_list)
```

6) The cube of a number is the number multiplied by itself 3 times. For example, $2^3 = 8$ and $10^3 = 1000$. In Python we write $2**3$ and $10**3$ to get the computer to compute these cubes. Define a function called cube which cubes each number in an input list and then returns the message "That was easy!" Test out your function on the list we defined earlier.

```
#Your answer to 6)
```

Exercise

Exercise

Define a new list called list1 which has all the even numbers between 0 and 10. Then define a function called addone which adds one to each number in list1 and prints "Mission Accomplished!" when done. Test that your function does what it is supposed to do.

2.3.7 if conditional statements

Note

In this section, we will introduce an if conditional statement as another useful tool in its own right and in writing user-defined functions. Basically a command(s) is executed only if a specified condition is true. If not, there is an 'else' option to specify a different command(s) to be executed.

1) Let's start by importing the numpy library. (Don't forget to press run (or shift+enter) to execute each cell.)

```
import numpy as np
```

2) Let's create a list of all numbers between 1 and 10.

```
list=np.arange(1,11,1)
list
```

3) Let's define a function checksize which takes an input list and outputs whether each number in the list is less than 7. If so, the computer will print "OK" and if not, the computer will print "Too Big!"

```
def checksize(list):  #remember the use of :
    for i in list:  #the for statement needs a : at the end. The next line  must also␣
↪be indented.
        if i<7:     #an if statement also needs a : at the end;  must also indent the␣
↪next line
            print(i,"is OK")    # computer does this if i<7
        else:       # else needs a : at the end; must indent the next line
            print(i, "is Too Big!") # instruction if x >=7
    return print("Finished!")  #prints a message to the screen when completed
```

4) Let's see if this works.

```
checksize(list)
```

5) Create a new list called list1 consisting of numbers from 1 to 20. Define a function numberdigits(list1) which goes through the numbers in list1 and prints "is single digit" if the number is single digit and "is double digits" if the number is double digit. Check that your function runs correctly on list1.

```
#Your answer to 5)
```

```
#test your anser to 5)
```

Exercise

Exercise

Define a new list called list2 which has all the even numbers between 0 and 20. Then define a function halve_upper_half(list2) which outputs half of each number in list2 if the original number is greater than 10. Check that your function does what it is supposed to do.

2.3.8 dataframes

Note

One important use of dataframes is the analysis of real world data. For such analysis, the basic Python skills we have introduced so far are all utilized:

- (user-defined) functions
- the numpy library
- the pandas library
- the matplotlib library
- for loops
- if conditionals

We will use a dataframe called COVID to explore COVID-19 data imported directly from the City of Chicago's Data Portal.

1) Let's start by importing the numpy and pandas libraries. In general, we will always begin by importing these two libraries (Don't forget to press run or shift+enter to execute each cell.)

```
import numpy as np
import pandas as pd
```

2) We can use pandas (pd) to get up-to-date info about COVID 19. Let's create a dataframe called COVID with this info and display the first line.

```
COVID=pd.read_json("https://data.cityofchicago.org/resource/yhhz-zm2v.json?
↪$limit=5000000")
COVID.head(1)
```

3) Let's list just the columns in the COVID dataframe.

```
COVID.columns
```

4) Let's get the number of rows and columns in our dataframe.

```
COVID.shape
```

5) Let's use just 4 columns: deaths_cumulative, population, tests_cumulative, and zip_code.

```
COVID=COVID[["deaths_cumulative", "population", "tests_cumulative","zip_code"]]
COVID.head(1)
```

6) Let's shorten the column names.

```
COVID.columns=["deaths","population","tests","zip"]
COVID.head(15)
```

7) We can get the latest test info for zip 60601 by first creating a datframe df for that zip code, and then using max() to get the highest value in the "tests" column.

```
df = COVID[COVID["zip"]=='60601']
numtested=df["tests"].max()
numtested
```

8) Let's define a function MyCOVID(COVID,zip) which allows us to enter a 5-digit zip code number and have the computer tell us how many tests, and the number of deaths.

```
def MyCOVID(COVID,zipcode):
    alreadychecked=0  #eliminate duplication of information
    for z in COVID.index:  #go through all the index values
        if COVID.loc[z,"zip"]==zipcode and alreadychecked==0:    #found the zip we␣
 ↪requested (first-time)
            alreadychecked=1  #we will only do this once
            df=COVID[COVID["zip"]==zipcode]
            numtested=df["tests"].max()
            numdeaths=df["deaths"].max()
            print("Zip code: ", COVID.loc[z,"zip"])
            print("number tested is ", numtested)
            print("number deaths ", numdeaths)
    return ("Enter a different zip code if you wish.")
```

9) Let's see if there's data for zipcode='60623'.

```
zipcode='60623'
MyCOVID(COVID,zipcode)
```

10) Now analyze zipcode='60637'

```
# Your answer to problem 10)
```

Exercise

Exercise

Modify the MyCOVID function so that a function MyCOVID2 also includes the population of the input zipcode. Then check that your function works on zipcode='60637'

2.4 Glimpse of Chicago

Note

In the Summer of 2020, Hope Wood (Wheaton College Urban Studies staff), Demetrius Crawford (Wheaton College Urban Studies outreach program participant) and I (Dr. Paul Isihara, Wheaton College math professor) launched the idea of creating JNB labs as a tool to teach math, data analysis and STEAM to disadvantaged Chicago grade-school students.

In Fall 2020, Mr. Jim Wilkes, principal of Cornerstone Academy, an alternative Christian High School serving mainly low to middle income students on the West Side of Chicago held the pilot offering of a JNB after-school program via Zoom. A class of Wheaton College student mentors worked online in small groups of 2 or 3 with Cornerstone students. Mr. Wilkes and I served as co-program directors, and Hope Wood and Antoinette Ratliff as program staff.

The following JNBs were developed for use with 8th graders participating in an after-school program at a non-profit organization called Celestial Ministries. The first two years, the scholars' middle school math teacher brought them to to the program and participated in the class. I served as teacher. Our goal was for the middle school teacher to eventually take over the primary teaching responsibilities.

Celestial Ministries serves mainly single parent children in the Lawndale neighborhood on the West Side of Chicago. Parents and friends were invited to a recognition ceremony in which students demonstrated various functionalities of JNBs which were covered in the program. Students who successfully completed the program were given a certificate and a 100 dollar stipend.

Word of Advice

There are likely opportunities to introduce Python Jupyter Notebooks in disadvantaged communities within proximity of your workplace or residence. Establishing a good working relationship with a well-established community organization and local community math teacher is highly recommended as opposed to seeking to establish a program on your own.

2.4.1 Michael Jordan's Greatest Scoring Game

Note

In this section, we will introduce a very important programming tool called a function.

Note

Please visit the web version of this section to engage with this interactive resource.

STEP#1 Review the PowerPoint on this lesson: link to PPT

STEP #2 From the PowerPoint, a variable is like a b ____ . A variable has a n _____ . We can store some i _____ such as a number (eg. 3) or a message (eg. 'God is Good!') in a variable. Let's give names to two variables:

- feet
- inch

In Michael Jordan's case, the info stored in the variable feet is the number 6 and the info stored in the variable inch is also equal to 6. For short, we just write feet=6 and inch=6 Let's go around and each person state your name and what your value is for the variables feet and inch.

(Answers: box, name, info)

STEP #3 The word for a short computer program is a c _____. The programming language we are using is Py _____ n. In the next cell, we give Python code to create the function in STEP 2. (If it looks complicated, try to find someone to help explain it to you.)

(Answers: code, Python)

```
def height(feet,inch):
    height_in_inches = 12*feet + inch
    return height_in_inches
```

```
height(6,0)  #Test on Seimone's height  feet=6, inch=0
```

```
72
```

STEP #4 What do we have to do in the next cell to test our function on 6'6" Michael Jordan?

```
# Test our function on Michael Jordan's height.
```

STEP #5 One of the fun parts of programming is that we can try to design a function to do almost anything we want.

Define a function called points(fg,ft,tp) which computes the points scored by a basketball player given the input variables

- fg= number of 2 point field goals made
- ft=number of free throws made
- tp=number of three point shots made

Use a variable called pts to store the answer.

STEP 6 In his highest scoring game, Michael Jordan made 21 two-point field goals, 21 free throws and 2 three point shots. Check that your function gives the correct value of 69 for his point total.

```
#Supply the missing values for points(fg,ft,tp)
```

For Discussion

1. What would you like your mentors to explain about today's lesson?
2. What is one of your favorite sports memories?

2.4.2 How Bad is COVID-19?

> **Note**
>
> In this section we will analyze data imported directly from the Chicago Data Portal
>
> https://data.cityofchicago.org/

STEP 1 Let's read in up to 100,000 rows of COVID data for Chicago from the Chicago Data Portal.

Let's check the number of rows and columns.

```
rawCOVID.shape
```

We'll list the 26 column names

```
list(rawCOVID.columns)
```

Q1 Does case matter when referring to a column name?

STEP TWO Let's streamline the data to just 4 columns, drop rows with missing data, simplify the column names, and then display the first 20 rows.

```
COVID=rawCOVID[['deaths_cumulative', "population",  'tests_cumulative','zip_code']]
COVID.dropna  #drop rows with missing data
COVID.columns=["deaths","population","tests","zip"] #simplify the column names
COVID.head(20)
```

Q2 Why is there different information for the same zip code?

STEP THREE

Let's check how many rows have data in each column.

```
COVID.count()
```

Q3 How many data rows are there in each column?

STEP FOUR

Let's find out how many times each zip code appears.

```
COVID["zip"].value_counts()
```

STEP FIVE

Let's make a copy of the COVID dataframe and display the first 5 rows.

```
df1=COVID
df1.head(5)
```

STEP SIX

Filter data by a specific zip code, for example, '60611'

```
df2=df1[df1['zip']=='60611']
df2.head()
```

```
df2["zip"].value_counts()
```

STEP SEVEN

Let's find the largest value in the 'tests' column for zip '60611'.

```
df2 = df1[df1["zip"]=='60637'] #get just rows with zip 60611
numtested=df2["tests"].max() #get the largest number for tests
numtested
```

Q4 How can we find how many deaths due to COVID have occurred in zip '60623'? (Create a new dataframe for 60623 called temp)

Q5 How can we get the population in 60623?

```
df3 = COVID[df1["zip"]=='60623'] #get just rows with zip 60611
pop=df3["population"].max() #get the largest number for tests
pop
```

STEP EIGHT We can instruct the computer to give us the number of COVID tests for any Chicago zipcode.

```
def MyCOVID(COVID,zip):
    alreadychecked=0  #eliminate duplication of information
    for z in COVID.index:  #go through all the index values
        if COVID.loc[z,"zip"]==zip and alreadychecked==0:    #found the zip we␣
↪requested (first-time)
            alreadychecked=1  #we will only do this once
            df=COVID[COVID["zip"]==zip]
            numtested=df["tests"].max()
            print("Zip code: ", zip)  #print zipcode
            print("number tested is ", numtested) #print number tested
    return ("Enter a different zip code if you wish.")
```

Q6 Test out the function on zip code '60610'

Exercises

Exercises

1. Define a function myCOVID2() which outputs for each given zip code the population, number tested, and number of deaths.
2. Use your function myCOVID2() to determine the COVID data for each of the following Chicago landmarks:

a) North Park University (zip '60625')

b) Wheaton in Chicago (zip='60637')

3. Why is COVID disproportionately impacting black and brown communities?

2.4.3 Mapping Famous Chicagoans

> **Note**
>
> In the section we will learn how to use Python to put information on a map of a geographical location.

Step One Let's begin by importing data of some famous people born in Chicago.

```
import pandas as pd
Chi=pd.read_csv('chicagoans.csv')
Chi
```

Q1 How old was Bobby Fischer when he became a grandmaster?

STEP TWO Let's use a Python library called folium to make a map of Chicago.

```
!pip install folium
```

```
import folium # map rendering library
from folium.features import DivIcon #used to add popup info to a map
Chicago_map = folium.Map(location=[41.886456, -87.62325],  tiles="openstreetmap",␣
↪zoom_start=10)
Chicago_map
```

> **Note**
>
> Please visit the web version of this section to engage with this interactive resource.

Q2

Try adjusting the "zoom_start" value. What happens?

What happens if you change the numbers in location=[41.886456, -87.62325]"

STEP THREE Let's add our data about famous Chicagoans to the map.

```
Chicago_map=folium.Map(location=[41.886456,-87.62325],tiles="openstreetmap",zoom_
↪start=11)
for i in Chi.index:
    p=[Chi.loc[i,"Lat"],Chi.loc[i,"Lon"]]
    folium.Marker(p,icon=DivIcon(
        icon_size=(100,0),
        icon_anchor=(0,8),
        html='<div style="font-size:20pt; color:red">'+str(Chi.loc[i,"Name"]) +'</div>
↪',
            )).add_to(Chicago_map)
    Chicago_map.add_child(folium.CircleMarker(p, radius=1,color='black'))
Chicago_map.save("Chicagoans.html")
Chicago_map
```

Q3 What side of the city was Michelle Obama born?

STEP FOUR Let's make a function which can add another person to our dataframe called Chi

Q4 What command is used to add a new line to the dataframe df?

STEP FIVE Let's add Jenifer Hudson to our dataframe using the following info:

'Jennifer Hudson',38,'yes','Actress','Englewood',60621,41.779699,-87.633194,'worked at Burger King'

Note

Please visit the web version of this section to engage with this interactive resource.

```
[Chi,Chicago_map]=addperson(Chicago_map,Chi,'Jennifer Hudson','Actress',41,'yes',
↪'Englewood',60621,41.779699,-87.633194,'worked at Burger King')
Chicago_map
```

Q5 Where did Jennifer Hudson work before becoming famous?

STEP SIX Add your info to the map by editing the info in the next cell.

```
#answer to Step Six
```

STEP SEVEN Use the following two commands to save your map and info in a new Excel file:

Chicago_map.save("Chicagoans.html")

Chi.to_excel('ChiRevised.xlsx')

Enter and run the commands in the cell below.

Q7 Did your map display correctly? Did the new Excel file include your name?

Discussion

1) Whom do you admire? Why so?
2) What character qualities can help someone to be successful in life?
3) What can help you to be more successful?

2.4.4 Predicting Exemplary Schools

Note

This section is a friendly competition predict whether a K-8 Chicago school's IL State Board of Education Summative Designation is designated 'Exemplary' (the 'Non-Exemplary' school categories are "Commendable", "Targeted", and "Comprehensive").

The prediction will be based on the file MiddleSchool.xlsx with the following data:

- Student Enrollment - Black or African American
- Student Enrollment - Hispanic or Latino
- Student Enrollment - Children with Disabilities
- Student Enrollment - Low Income
- Total Number of School Days
- 8th Grade Passing Algebra 1
- Student Attendance Rate

- Student Chronic Truancy Rate
- Avg Class Size – All Grades
- Teacher Retention Rate

Scoring

Scoring for our competition is based on values for the **Confusion Matrix**:

$$\begin{pmatrix} TP & FN \\ FP & TN \end{pmatrix}$$

where

- TP=True Positive: your model predicts exemplary and the school is exemplary
- TN=True Negative: your model predicts not exemplary and the school is not exemplary
- FP=False Positive: your model predicts exemplary but the school is not exemplary
- FN=False Negative: your model predicts not exemplary but the school is exemplary

The number of each type of prediction then determines

- **Accuracy** = $\frac{|TP|+|TN|}{|TP|+|TN|+|FP|+|FN|}$ (proportion that were correctly predicted out of all the schools)
- **Specificity (Precision)** = $\frac{|TP|}{|TP|+|FP|}$ (proportion that were correct out of those you predicted to be exemplary)
- **Sensitivity (Recall)** = $\frac{|TP|}{|TP|+|FN|}$ (proportion that you predicted correctly among just the exemplary schools)

Your competition (F1) score is the geometric mean of the precision (specificity), and recall(sensitivity):

$$F_1 = 2\frac{precision * recall}{precision + recall}$$

STEP ONE Exploratory Data Analysis 1a) Import the usual libraries including matplotlib.pyplot as plt, as well as the MiddleSchool report card data into a datframe df.

1b) Display the column names and the number of data rows in each column.

```
df.count()
```

1c Get the max and min values in each column.

1d Check how many schools are in each category.

```
df["Summative Designation"].value_counts()
```

STEP TWO Define a function which predicts whether a school is exemplary (1) or not-exemplary (0).

2a For a simple prediction, let us predict that a school is exemplary if the Teacher Retention Rate is at least 90%.

Assignment

Assignment

Modify the Prediction Model to see how high you can score.

2.4.5 The Great Migration

DATASETS:

migration.xlsx (source https://www.census.gov/dataviz/visualizations/020/508.php); uscities.xlsx ((https://simplemaps.com/data/us-cities))

SUMMARY: We create a heat map to show the %change in African Americans living in cities in the Northern and Southern USA as a result of the Great Migrations. Those with the largest increase (over 20%) were all in the North.

BACKGROUND: Isabel Wilkerson gives a good introduction to the Great Migration. (Run the next cell)

"The Great Migration and the power of a single decision | Isabel Wilkerson" YouTube, uploaded by TED, 6 April 2018, https://youtu.be/n3qA8DNc2Ss?si=kF0l5_YhY-cmmLfO. Permissions: YouTube Terms of Service

Note

Please visit the web version of this section to engage with this interactive resource.

```
from IPython.display import YouTubeVideo
YouTubeVideo('n3qA8DNc2Ss')
```

```
### Creating a Dataframe with %Changes in African American Population by (Lat,Lon) Location
```

1) Use pandas to import the datafile migration.xlsx into a dataframe called migration.

```
migration=pd.read_excel('migration.xlsx')
```

2) Display the first two rows of migration.

```
migration.head(2)
```

Shorten the column names (MIg1=1st Great Migration, Mig2=2nd Great Migration), separate the City and State, and create a multi-index with city and state.

4) Read the lat lon data for US cities in the file "uscities.xlsx" and make ["city","state"] the multi-index.

5) We'll perform an inner join (intersection) of the two dataframes migdf and latlon. This will add the lat lon data to the migdf data

```
df=pd.merge(latlon,migdf, how='inner', left_index=True,right_index=True)
df.columns=["lat","lon","Mig2","Mig1"]
df.to_excel("GM.xlsx")
df.head(5)
```

6) Check for missing data.

```
df[df["Mig1"]=="No data"].count()
```

```
df[df["Mig2"]=="No data"].count()
```

7) Remove rows with missing data.

```
df.count()
```

```
df=df[df["Mig1"]!="No data"]
df=df[df["Mig2"]!="No data"]
df.count()
```

Creating a Map from our Dataframe

8) Make a heat map of the 1st Great Migration.

9) Improve clarity by specifying certain cities to display.

ASSIGNMENT

Assignment

Make heatmaps for Mig2 and compare it with the heatmaps for Mig1. (Eg. What happened in Chicago?)

```
#import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt # plotting library
import sklearn
from sklearn.cluster import KMeans
```

2.4.6 The Chicago Hardship Index

Note

This section is all about Chicago, so has been included here. As it contains more advanced coding methods (explained in later chapters), the coding cells may be skimmed or skipped entirely and returned to later. The contents first appeared in an open access article distributed under the terms of the Creative Commons CC BY license:

Amdat, W. C. (2021). The Chicago Hardship Index: An Introduction to Urban Inequity. Journal of Statistics and Data Science Education, 29(3), 328–336. https://doi.org/10.1080/26939169.2021.1994489

Raw Data and Summary Statistics

The hardship index is a way to use data to explore urban inequities. Exploratory data analysis for social justice issues should aim to be accessible, significant, equitable, impartial, and transparent.

Criteria	Assessment question
(i) Accessible	Is the data presentation understandable to key stakeholders, including policymakers and community leaders?
(ii) Significant	Is the information relevant and accurate, and hence of enduring value to problem solvers and decisionmakers?
(iii) Equitable	Does the data gathered adequately encompass the needs of the vulnerable who are often overlooked?
(iv) Impartial	Is all the data in hand fairly represented, especially outliers which may call into question the status quo?
(v) Transparent	Are the underlying methods sufficiently clear that others can reproduce/confirm the results?

The hardship index (HI) is the mean of six indicator estimates. Each indicator has been normalized and scaled from 0-100 (except per capita income). A higher hardship index value indicates greater hardship. The 6 raw indicator estimates are complicated by time variation over 5-year periods (US Census Bureau 2008) and geographical boundary approximation (Great Cities Institute 2019). The indicators are

- UNEMP = % of community age 16 and older who are unemployed
- NOHS = % of community age 25 and older without a high school diploma
- DEP = % of community who are dependent (under age 18 or over age 64)
- HOUS= % of community with overcrowded housing (more than 1 occupant per room)
- POV = % below federal poverty line
- INC = per capita income

Datafile: 'HIHOM20142017.xlsx'

Data Source: https://greatcities.uic.edu/wp-content/uploads/2016/07/GCI-Hardship-Index-Fact-SheetV2.pdf (2010-2014) https://greatcities.uic.edu/wp-content/uploads/2019/12/Hardship-Index-Fact-Sheet-2017-ACS-Final-1.pdf (2013-2017).

Summary Statistics

1) Upload the hardship index data for 2014 and 2017. The first two of seventy-seven community areas in Chicago are printed below.

```
raw_hardship=pd.read_excel('HIHOM20142017.xlsx')
raw_hardship.head(2)
```

2) Separate the 2014 and 2017 hardship index (HI) data into two dataframes called "dfHI14" and "dfHI17." The column names will reflect the year

```
dfHI14=raw_hardship[["Community","HI14","UNEMP14","NOHS14","DEP14","HOUS14","POV14",
 ↪"INC14"]]
dfHI14 = dfHI14.rename(columns = {'Community':'Community14'})
dfHI17=raw_hardship[["Community","HI17","UNEMP17","NOHS17","DEP17","HOUS17","POV17",
 ↪"INC17"]]
dfHI17 = dfHI17.rename(columns = {'Community':'Community17'})
```

3) The first two communities' HI and the indicator scores for 2014 are printed below.

```
dfHI14.head(2)
```

4) The scores for two communities in 2017 are printed below.

```
dfHI17.head(2)
```

5) Calculate the summary statistics for indicators from the data in all seventy-seven communities. For example, we calculate statistics for the HOUS indicator in 2017. By replacing "HOUS17," statistics for other indicators can be calculated–like "HI17," "NOHS17", etc.

```
x=dfHI17["HOUS17"]
import numpy
print("Minimum: ", numpy.min(x))
print("Maximum: ", numpy.max(x))
print("Standard Deviation: ", numpy.std(x))
print("Mean: ", numpy.mean(x))
print("Median: ", numpy.median(x))
```

The table below lists the indicator values for several communities, as shown above, and the summary statistics for each indicator.

Community area	Hardship Index Score	Percent of crowded housing	Percent of households with income below poverty level	Unemployment rate for population Age 16 and over	Percent Age 25 and older with no high school diploma	Percent under Age 18 or over Age 64	Per capita income
1. Rogers Park	39.4	6.6	22.6	7.7	9.5	26.2	$25,874
25. Austin	53.1	4.4	28.1	15.4	13.6	38.6	$17,586
32. Loop	9.0	2.8	11.3	4.6	1.4	16.7	$77,722
41. Hyde Park	25.3	1.6	23.3	6.7	2.2	26.4	$41,719
42. Woodlawn	50.4	3.5	35.2	16.9	8.2	36.7	$19,117
68. Englewood	70.5	4.0	42.4	34.1	16.1	41.5	$12,777
77. Edgewater	28.9	4.1	15.8	6.2	7.3	25.2	$38,748
min	8.6	0.3	4.3	2.9	1.4	16.7	$9,601
max	84.2	13.6	58.8	35.6	28.8	47.6	$91,948
std	16.5	2.9	10.9	7.9	6.2	6.4	$17,303
mean	43.7	4.1	21.1	12.7	11.2	35.4	$29,016
median	45.7	3.3	20.2	10.8	9.7	37.5	$23,990

K-Means Clustering

We will use a machine learning method called K-means clustering (see Chapter 10.3) to visualize patterns in the geographic information. By plotting the homicide data with it, we can investigate a relationship between location, hardship cluster, and number of homocides. We use the sklearn library for machine learning.

Datafile: 'standardizedindicators.xlsx'

1) Read standardized hardship index and homicide data.

```
hom_df = pd.read_excel('standardizedindicators.xlsx')
hom_df.head(2)
```

2) Create dataframe with just HI and HOM 2017 info

```
HIHOM=hom_df[["UNEMP17","NOHS17","DEP17","HOUS17","POV17","INC17"]]
HIHOM.head()
```

3) Use the KMeans() function to make n_clusters=2 clusters and get the labels indicating which cluster each point belongs to.

```
# Fit the k means model
k_means = KMeans(init="k-means++", n_clusters=2, n_init=2)
k_means.fit(HIHOM)
#Get Labels
k_means_labels = k_means.labels_
k_means_labels
```

4) Add the labels to hom_df

```
hom_df["CLASS"]=k_means_labels
hom_df.head(2)
```

5) Make a geographic plot of Chicago's 77 community areas with marker color (blue or red) indicating k-means bi-clustering based only on the 6 standardized economic hardship indicators. Affluent community areas such as the Loop (central business district), Near North Side (which includes the "Gold Coast"), and Hyde Park (site of the University of Chicago) appear in blue. Lower-income communities with a history of injustices, including Woodlawn, Englewood, and Austin, appear in red. Marker sizes are proportional to homicide counts, with actual numbers in parentheses following named community areas.

Exercise

Note that the Near North Side has the same homicide count as Riverdale. However, the homicide rate in Riverdale was 10 times higher than the Near North Side. Using the 2017 data, change the marker size so that it corresponds to homicide rate (per 10,000 people) rather than homicide count.

2.5 Arts in STEM (STEAM)

Note

The inclusion of arts in STEM is a good way to engage students with diverse interests and abilities. In this chapter, we offer a few examples of a fun way to think about mathematical and programming concepts.

2.5.1 Pixel Images

Note

In this section, we will show how increasing the number of pixels used to represent an image will sharpen the resolution.

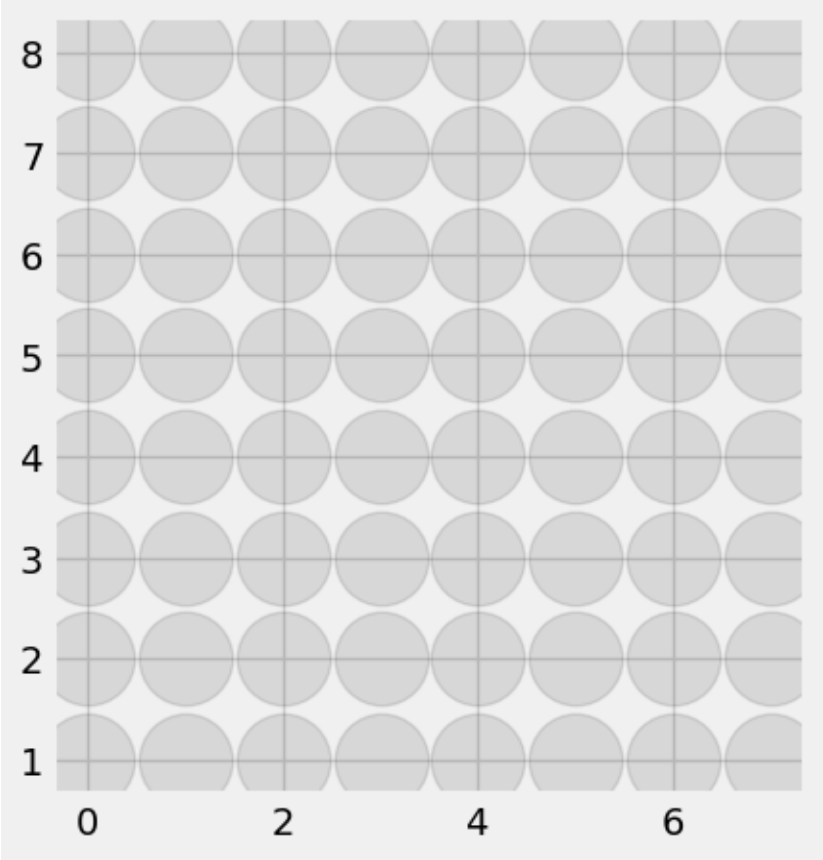
1. Alpha Values The opacity of an image can be adjusted using an alpha value between 0 and 1. For example, a black circle with alpha equal to .1 appears as a light gray and is black when alpha is equal to 1. Note that a rectangular array of alpha values can then be mapped to a rectangular array of circles which have the corresponding shading.

```
xls=pd.ExcelFile('data.xlsx')
num=3 #number of images
N=8  #NxN pixels
character_collection = {}
for i in np.arange(0,num):
    character_collection[i]=pd.read_excel(xls,str(i))
```

```
character_collection[0]
```

	0	1	2	3	4	5	6	7
0	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
3	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
4	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
5	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
6	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
7	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

```
ig=plt.figure(figsize=(4.5,5))
for i in np.arange(0,N):
    for j in np.arange(0,N):
        plt.plot(j,8-i, marker="o", markersize=35, markeredgecolor='k',␣
↪markerfacecolor="k",alpha=character_collection[0].loc[i,j])
```




```
character_collection[1]
```

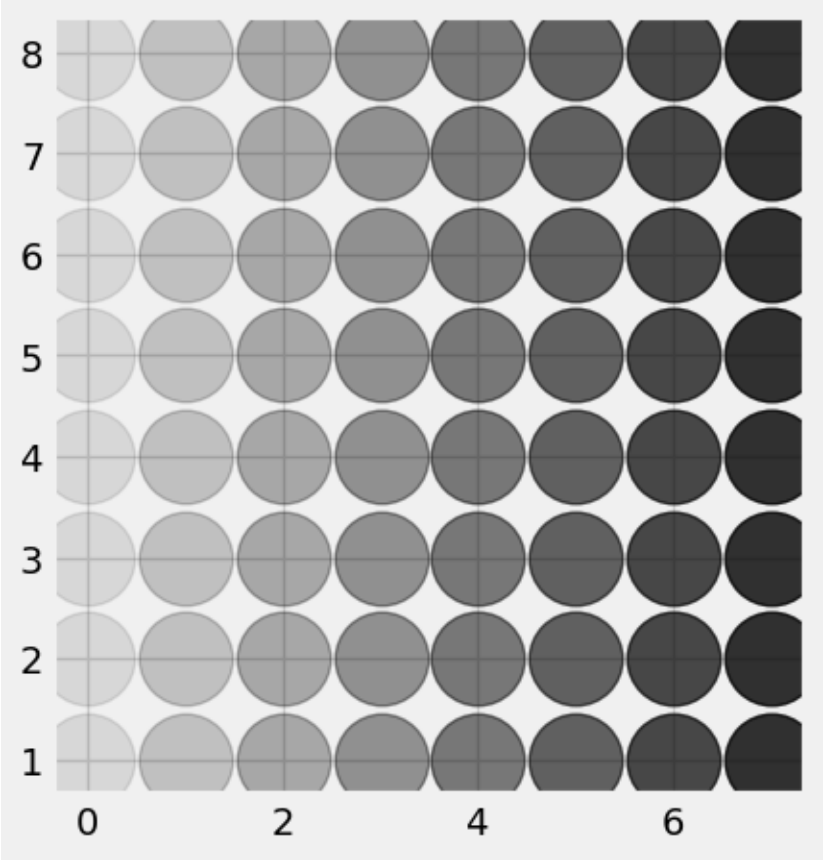
	0	1	2	3	4	5	6	7
0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
1	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
3	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
4	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8

(continues on next page)

(continued from previous page)

```
5  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8
6  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8
7  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8
```

```
fig=plt.figure(figsize=(4.5,5))
for i in np.arange(0,N):
    for j in np.arange(0,N):
        plt.plot(j,8-i, marker="o", markersize=35, markeredgecolor='k',␣
↪markerfacecolor="k",alpha=character_collection[1].loc[i,j])
```




```
character_collection[2]
```

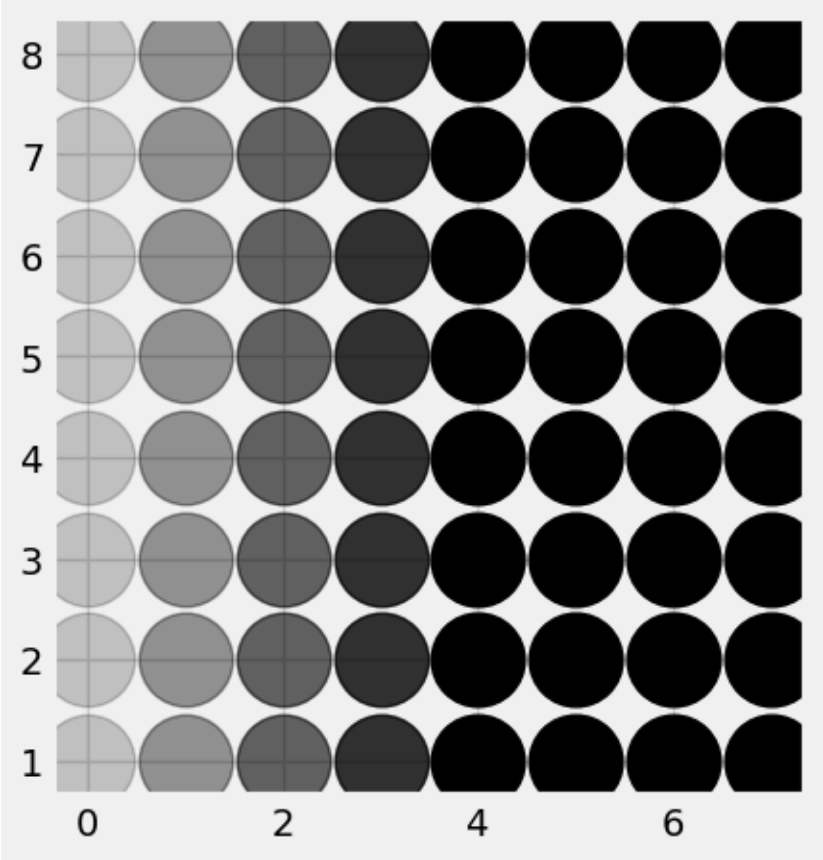
```
     0    1    2    3  4  5  6  7
0  0.2  0.4  0.6  0.8  1  1  1  1
1  0.2  0.4  0.6  0.8  1  1  1  1
2  0.2  0.4  0.6  0.8  1  1  1  1
3  0.2  0.4  0.6  0.8  1  1  1  1
4  0.2  0.4  0.6  0.8  1  1  1  1
5  0.2  0.4  0.6  0.8  1  1  1  1
6  0.2  0.4  0.6  0.8  1  1  1  1
7  0.2  0.4  0.6  0.8  1  1  1  1
```

```
fig=plt.figure(figsize=(4.5,5))
for i in np.arange(0,N):
    for j in np.arange(0,N):
```

(continues on next page)

(continued from previous page)

```
        plt.plot(j,8-i,marker="o", markersize=35, markeredgecolor='k',␣
↪markerfacecolor="k",alpha=character_collection[2].loc[i,j])
```



2. Pixel Images

We can use Python to create NxN pixel images of a given photo.

For a given black and white picture, a computer code will create an $N \times N$ array of numbers with gray-scale values.

Or, given an $N \times N$ array of gray scale values, another computer code can reconstruct a pixel image.

First we install a library for working with pixel images.

```
!pip install opencv-python
```

```
Collecting opencv-python
  Downloading opencv_python-4.8.0.74-cp37-abi3-win_amd64.whl (38.1 MB)
                                              0.0/38.1 MB ? eta -:--:--
                                              0.1/38.1 MB 3.5 MB/s eta 0:00:11
                                              0.5/38.1 MB 6.7 MB/s eta 0:00:06
     -                                        1.0/38.1 MB 7.8 MB/s eta 0:00:05
     -                                        1.4/38.1 MB 7.6 MB/s eta 0:00:05
     --                                       1.9/38.1 MB 8.7 MB/s eta 0:00:05
     --                                       2.4/38.1 MB 8.6 MB/s eta 0:00:05
     ---                                      2.9/38.1 MB 9.2 MB/s eta 0:00:04
     ---                                      3.3/38.1 MB 9.2 MB/s eta 0:00:04
     ----                                     3.9/38.1 MB 9.5 MB/s eta 0:00:04
     ----                                     4.3/38.1 MB 9.5 MB/s eta 0:00:04
```

(continues on next page)

(continued from previous page)

```
-----                                      4.9/38.1 MB 9.7 MB/s eta 0:00:04
-----                                      5.4/38.1 MB 9.8 MB/s eta 0:00:04
------                                     5.9/38.1 MB 9.9 MB/s eta 0:00:04
------                                     6.4/38.1 MB 10.0 MB/s eta 0:00:04
-------                                    7.0/38.1 MB 9.9 MB/s eta 0:00:04
-------                                    7.5/38.1 MB 10.2 MB/s eta 0:00:03
--------                                   8.0/38.1 MB 10.1 MB/s eta 0:00:03
--------                                   8.5/38.1 MB 10.3 MB/s eta 0:00:03
---------                                  9.1/38.1 MB 10.2 MB/s eta 0:00:03
----------                                 9.5/38.1 MB 10.2 MB/s eta 0:00:03
----------                                10.1/38.1 MB 10.2 MB/s eta 0:00:03
----------                                10.5/38.1 MB 10.6 MB/s eta 0:00:03
-----------                               11.1/38.1 MB 10.7 MB/s eta 0:00:03
-----------                               11.7/38.1 MB 10.9 MB/s eta 0:00:03
------------                              12.2/38.1 MB 10.9 MB/s eta 0:00:03
-------------                             12.7/38.1 MB 10.9 MB/s eta 0:00:03
-------------                             13.2/38.1 MB 10.9 MB/s eta 0:00:03
--------------                            13.7/38.1 MB 10.9 MB/s eta 0:00:03
--------------                            14.3/38.1 MB 11.1 MB/s eta 0:00:03
---------------                           14.8/38.1 MB 11.1 MB/s eta 0:00:03
---------------                           15.4/38.1 MB 11.1 MB/s eta 0:00:03
----------------                          15.9/38.1 MB 10.9 MB/s eta 0:00:03
----------------                          16.4/38.1 MB 11.1 MB/s eta 0:00:02
-----------------                         16.9/38.1 MB 11.1 MB/s eta 0:00:02
-----------------                         17.4/38.1 MB 10.9 MB/s eta 0:00:02
------------------                        17.9/38.1 MB 10.9 MB/s eta 0:00:02
------------------                        18.5/38.1 MB 10.9 MB/s eta 0:00:02
-------------------                       19.0/38.1 MB 11.1 MB/s eta 0:00:02
-------------------                       19.5/38.1 MB 11.1 MB/s eta 0:00:02
--------------------                      20.1/38.1 MB 10.9 MB/s eta 0:00:02
--------------------                      20.5/38.1 MB 11.1 MB/s eta 0:00:02
---------------------                     21.0/38.1 MB 10.9 MB/s eta 0:00:02
---------------------                     21.5/38.1 MB 10.7 MB/s eta 0:00:02
----------------------                    22.0/38.1 MB 10.9 MB/s eta 0:00:02
----------------------                    22.4/38.1 MB 10.7 MB/s eta 0:00:02
-----------------------                   23.0/38.1 MB 10.9 MB/s eta 0:00:02
-----------------------                   23.5/38.1 MB 10.7 MB/s eta 0:00:02
------------------------                  24.0/38.1 MB 11.1 MB/s eta 0:00:02
------------------------                  24.5/38.1 MB 10.9 MB/s eta 0:00:02
-------------------------                 25.0/38.1 MB 10.7 MB/s eta 0:00:02
-------------------------                 25.4/38.1 MB 10.9 MB/s eta 0:00:02
--------------------------                25.9/38.1 MB 10.7 MB/s eta 0:00:02
--------------------------                26.5/38.1 MB 10.7 MB/s eta 0:00:02
---------------------------               27.1/38.1 MB 10.7 MB/s eta 0:00:02
---------------------------               27.5/38.1 MB 10.7 MB/s eta 0:00:01
----------------------------              28.1/38.1 MB 10.7 MB/s eta 0:00:01
----------------------------              28.6/38.1 MB 10.7 MB/s eta 0:00:01
-----------------------------             29.1/38.1 MB 10.6 MB/s eta 0:00:01
-----------------------------             29.6/38.1 MB 10.7 MB/s eta 0:00:01
------------------------------            30.1/38.1 MB 10.6 MB/s eta 0:00:01
------------------------------            30.7/38.1 MB 10.6 MB/s eta 0:00:01
-------------------------------           31.2/38.1 MB 10.9 MB/s eta 0:00:01
-------------------------------           31.7/38.1 MB 10.7 MB/s eta 0:00:01
--------------------------------          32.2/38.1 MB 10.7 MB/s eta 0:00:01
--------------------------------          32.7/38.1 MB 10.7 MB/s eta 0:00:01
---------------------------------         33.2/38.1 MB 10.9 MB/s eta 0:00:01
---------------------------------         33.7/38.1 MB 10.7 MB/s eta 0:00:01
```

(continues on next page)

(continued from previous page)

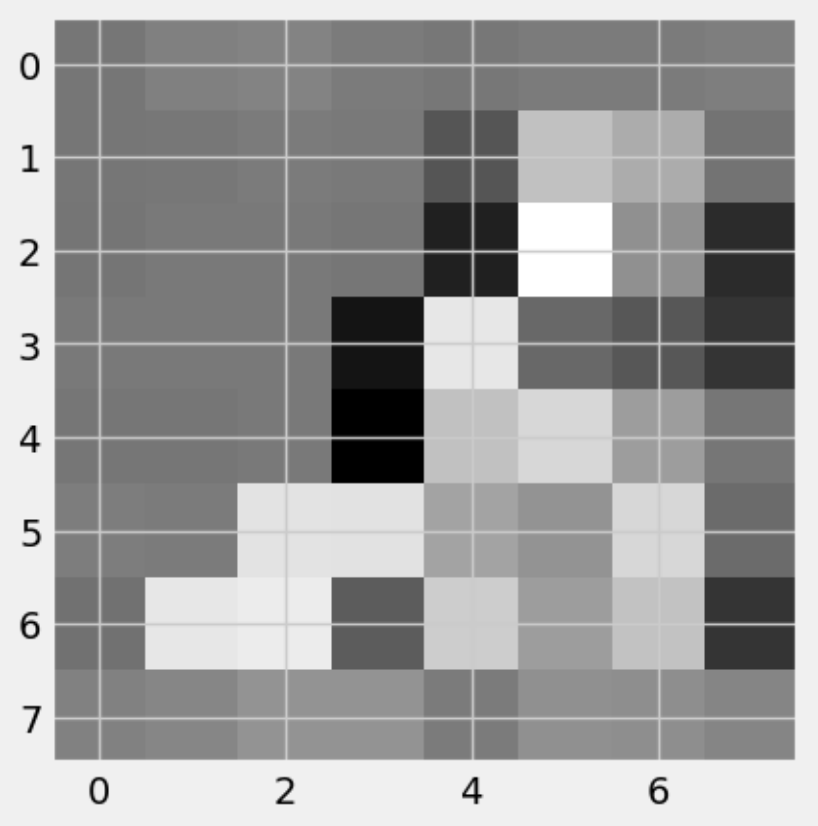
```
     ---------------------------------      34.1/38.1 MB 10.6 MB/s eta 0:00:01
     ---------------------------------      34.7/38.1 MB 10.7 MB/s eta 0:00:01
     ----------------------------------     35.2/38.1 MB 10.7 MB/s eta 0:00:01
     ----------------------------------     35.7/38.1 MB 10.9 MB/s eta 0:00:01
     -----------------------------------    36.2/38.1 MB 10.9 MB/s eta 0:00:01
     -----------------------------------    36.8/38.1 MB 10.9 MB/s eta 0:00:01
     ------------------------------------   37.3/38.1 MB 10.7 MB/s eta 0:00:01
     ------------------------------------   37.8/38.1 MB 10.9 MB/s eta 0:00:01
     ------------------------------------   38.1/38.1 MB 10.7 MB/s eta 0:00:01
     ------------------------------------   38.1/38.1 MB 10.7 MB/s eta 0:00:01
     ---------------------------------------- 38.1/38.1 MB 9.8 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.21.2 in c:\users\pisihara\appdata\local\
 ↪anaconda3\lib\site-packages (from opencv-python) (1.24.3)
Installing collected packages: opencv-python
Successfully installed opencv-python-4.8.0.74
```

The following function takes a picture file stored in a folder (in our case we will call the folder “images”) and creates an N x N pixel image.

First let’s create an 8x8 pixel image. Can you guess the original picture?

```
makepixelimage("images", 8)
```

```
<matplotlib.image.AxesImage at 0x1e18f71b3d0>
```



Assignment

Assignment

1a) By increasing the number of pixels, we can get a better reproduction of the original. Create a 16x16 pixel image.
b) Create a 32x32 pixel image.

2) Upload a different image into a folder called "my_images". Then use the makepixelimage(,) function to create an 8x8, 16x16 and 32x32 pixel image and see if the others in the class can guess your original image

2.5.2 Word Clouds

Note

In this lab, we will create **word clouds** – artistic representations of words from a speech, book, song or poem. The frequency or number of times a word occurs is represented by the size of that word in the word cloud.

Watch the famous "I have a Dream" speech by Dr. Martin Luther King

"Martin Luther King's final speech: 'I've been to the mountaintop'" YouTube, uploaded by Guardian News, 4 April 2018, https://youtu.be/e49VEpWg61M?si=2IYOozVbQUfv9SUH. Permissions: YouTube Terms of Service

```
from IPython.display import YouTubeVideo
YouTubeVideo('o8dzxh7Ybqw', width=800, height=300)
```

1. Import the Python data analytics (pandas or pd) and numerical Python (numpy) libraries.

Q1 What is the abbreviation for numpy?

2. Use pandas (pd) to read in the Excel file with the words of the chorus "I have the joy, joy, joy, joy, down in my heart."

```
  WORDS
0     I
1  have
2   the
3   joy
4   joy
```

Q2 What word is in row 4?

3. Get the frequencies of each word

```
joy      8
down     5
in       5
heart    5
my       3
where    3
I        2
have     2
the      2
my       2
to       1
```

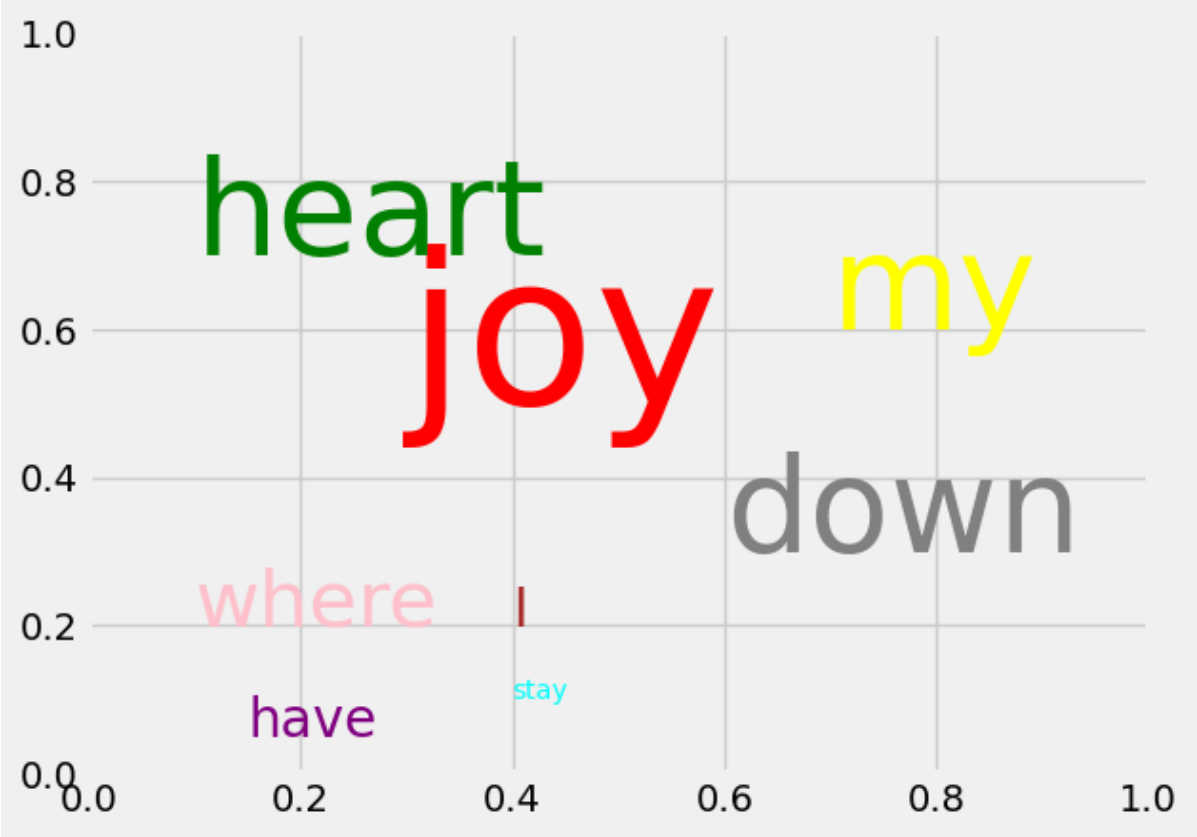
(continues on next page)

(continued from previous page)

```
stay     1
Name: WORDS, dtype: int64
```

Q3 Which word has the highest frequency?

4. We can make a simple word cloud by putting each important word in a specified place, with a given color, and given size. Add the text joy to the word cloud at position .3,.5 in color 'red' and fontsize=80.



Q4 What command is used to include the word joy?

5. Data scientists have created functions to make word clouds so we don't have to enter each word or letter one by one. These are found in the wordcloud library.

```
!pip install wordcloud
```

```
Collecting wordcloud
  Downloading wordcloud-1.9.2-cp311-cp311-win_amd64.whl (151 kB)
                                              0.0/151.4 kB ? eta -:--:--
     ------------------------------------- 151.4/151.4 kB 4.6 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.6.1 in c:\users\pisihara\appdata\local\
 ↪anaconda3\lib\site-packages (from wordcloud) (1.24.3)
Requirement already satisfied: pillow in c:\users\pisihara\appdata\local\anaconda3\
 ↪lib\site-packages (from wordcloud) (9.4.0)
Requirement already satisfied: matplotlib in c:\users\pisihara\appdata\local\
 ↪anaconda3\lib\site-packages (from wordcloud) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\pisihara\appdata\local\
 ↪anaconda3\lib\site-packages (from matplotlib->wordcloud) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\pisihara\appdata\local\
```

(continues on next page)

(continued from previous page)

```
 ↪anaconda3\lib\site-packages (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\pisihara\appdata\
 ↪local\anaconda3\lib\site-packages (from matplotlib->wordcloud) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\pisihara\appdata\
 ↪local\anaconda3\lib\site-packages (from matplotlib->wordcloud) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\pisihara\appdata\local\
 ↪anaconda3\lib\site-packages (from matplotlib->wordcloud) (23.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\pisihara\appdata\local\
 ↪anaconda3\lib\site-packages (from matplotlib->wordcloud) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\pisihara\appdata\
 ↪local\anaconda3\lib\site-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\pisihara\appdata\local\
 ↪anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib->wordcloud)␣
 ↪(1.16.0)
Installing collected packages: wordcloud
Successfully installed wordcloud-1.9.2
```

```
import wordcloud #used to make a wordcloud
```

Q5 What is the name of the library which creates word clouds?

6. Check how fast the computer can read the text file with the words to Dr. Martin Luther King's "I have a dream" speech.

```
0.009278297424316406
```

Q6 How fast did the computer read the text file?

7. Define a function to take out punctuation marks and count the interesting words

Q7 What uninteresting words does the above function remove from the word cloud?

8. Make the word cloud by executing the next cell. Then modify the calculate_frequencies() function to eliminate uninteresting words. Also change how many words are displayed.

Q8 What is the name of the file with the word cloud image for the "I have a dream" speech?

Assignment

> **Assignment**
>
> Create word cloud of a famous speech, poem, song etc. Then edit out all the uninteresting words. Then see if the others in the class can figure out your source from 10 words, 20 words, 30 words etc.

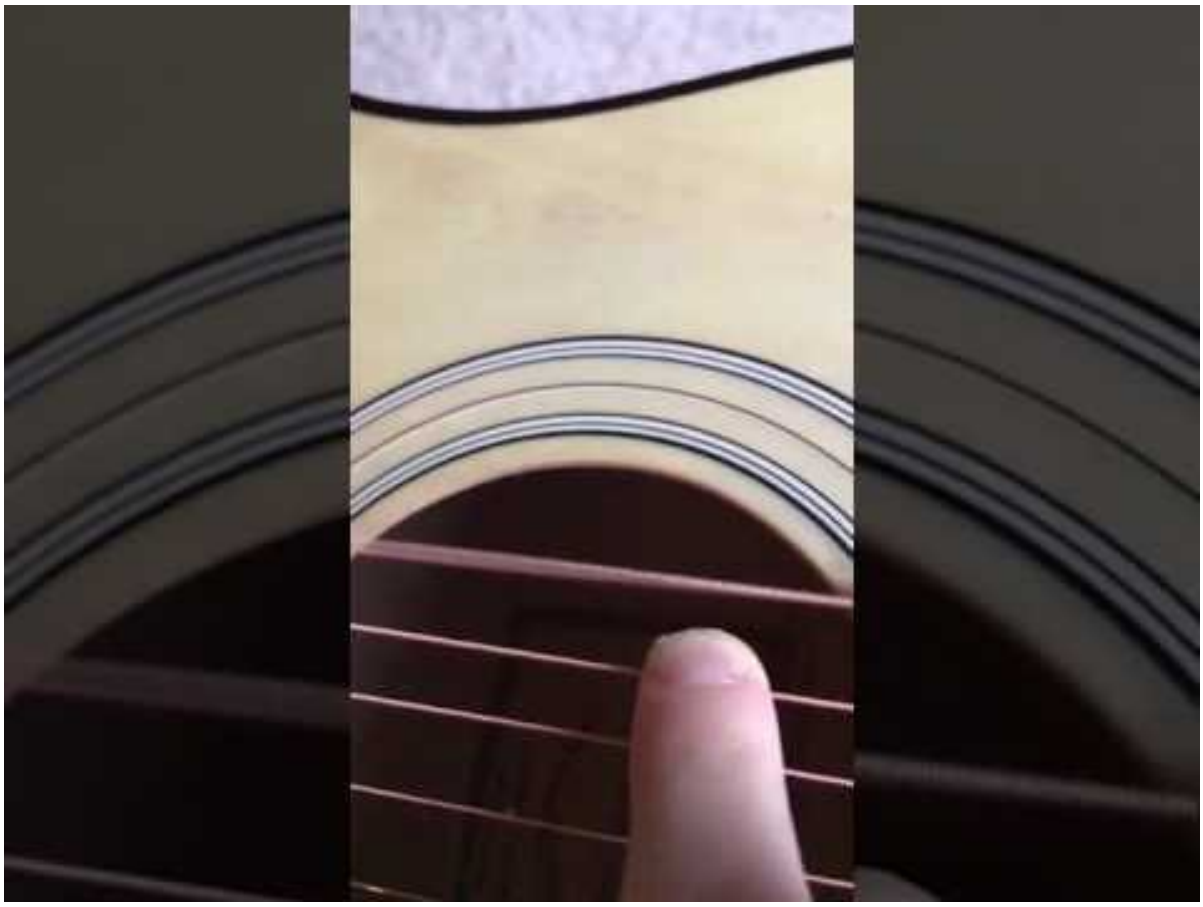
2.5.3 Name That Tune!

> **Note**
>
> In this lab we will explore the basic relationship between sound and a mathematical description of sound using sine waves.

1. Watch the video "Slow-motion Vibrating Guitar String!" YouTube, uploaded by LB's Epic Tutorials, 23 January 2017, https://youtu.be/tVYQRC1-D54?si=V9ahx88FVM8GXT2i. Permissions: YouTube Terms of Service

and observe that the periodic vibration of a guitar string causes a sound wave.

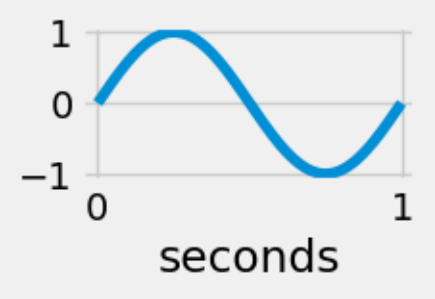
```
from IPython.display import YouTubeVideo
YouTubeVideo('tVYQRC1-D54')
```



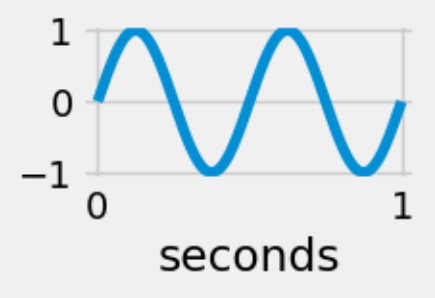
Pure tones can be modelled using sine waves. The amplitude of a sine wave (high high up and down it oscillates) is

proportional to the volume of the sound. The frequency of a sine wave is defined as the number of times a wave completes its up and down cycle in one second. The higher the frequency of a sound wave, the higher the pitch. Use the function defined in the next cell to check what happens to a sine wave if we change its frequency.

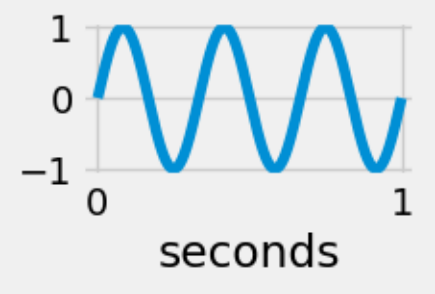
```
sinewave(1)  #frequency=1 and 1 cycle per second
```



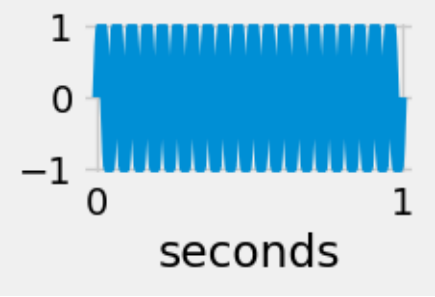

```
sinewave(2)  #frequency=2 and 2 cycles per second
```




```
sinewave(3) #frequency=3 and 3 cycles per second
```




```
sinewave(20) #frequency=20 and 20 cycles per second
```



2. If an object vibrates at high enough of a frequency, it creates a sound wave that we can hear. The following function converts a frequency into a sine wave and then into an audible sound

Note

Please visit the web version of this section to engage with this interactive resource.

```
play(220) # play a sound at 220 hz
```

```
<IPython.lib.display.Audio object>
```

Doubling the frequency creates the same note but an octave higher.

```
play(440) # play a sound at 440 hz
```

```
<IPython.lib.display.Audio object>
```

3. A musical scale can be created by multipling the first note in the scale by different fractions

The following function can play a scale or song for us.

```
play(scale)
```

```
<IPython.lib.display.Audio object>
```

```
play(twooctavescale)
```

```
<IPython.lib.display.Audio object>
```

4. We can create tunes by arranging notes in a scale.

```
weshallovercome = [so, so , la, la, so, fa,mi,rest,so, so , la, la, so, fa,mi,rest,so,
↪so,la,ti,do1,do1,re1,re1,ti,la,ti,la,so,so,la,ti,do1,do1,ti,la,so,so,rest,rest,la,
↪la,so,fa,mi,mi,rest,rest,so,so,do,fa,mi,mi,re,re,do,do,do,do,rest,rest]
```

```
play(weshallovercome)
```

```
<IPython.lib.display.Audio object>
```

Assignment

Assignment

Create your own tune. When you are finished, see if the others can name that tune!

2.5.4 Random Numbers and Pi

Note

In this lab we will use random numbers to estimate $\pi \approx 3.14$.

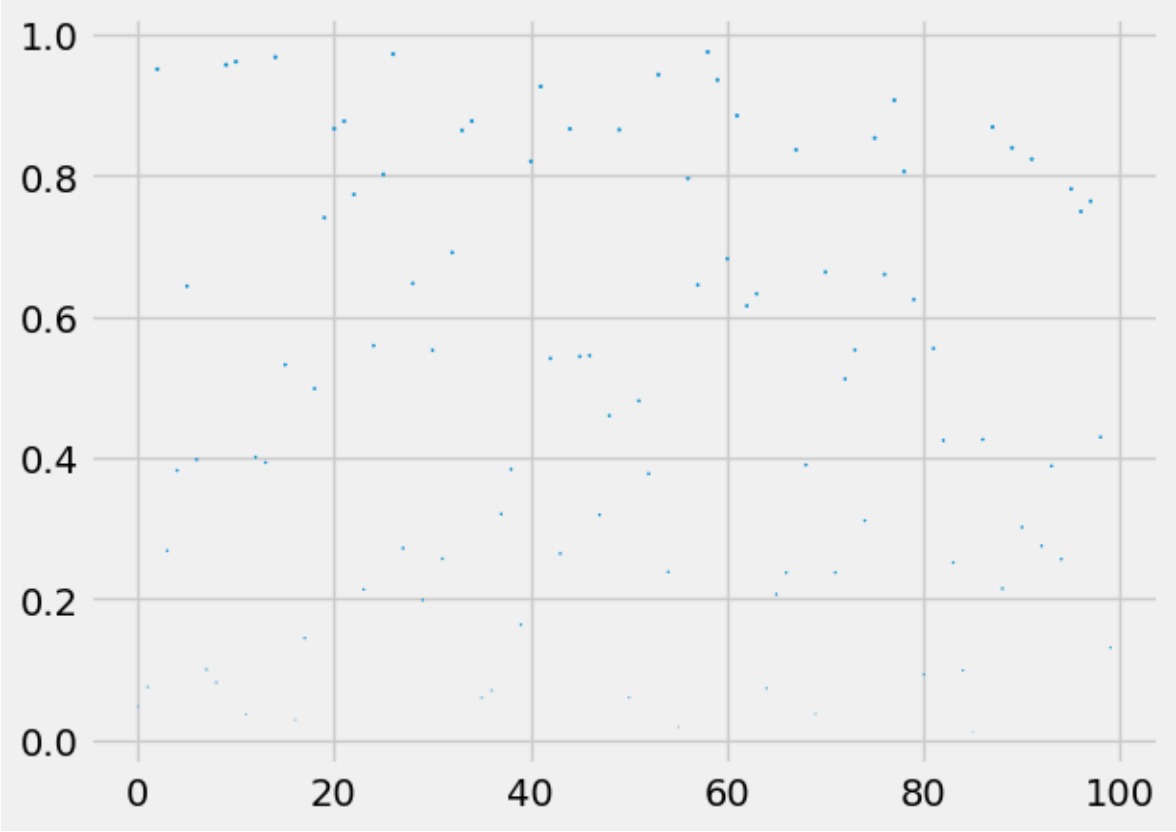
1) Let's look at the library matplotlib. The matplotlib (mpl) library has a sub-library called pyplot (plt). We can access the functions in pyplot by executing the next cell

```
import matplotlib.pyplot as plt
import numpy as np
np.random.seed(1232023)  # The seed determines a specific stream of pseuod  random␣
↪numbers
```

1. A random number is any number whose decimal is between 0 and 1. Let's use a random number generator to generate 100 random numbers and then plot them.

```
N=100
x=np.arange(0,N,1)
y=np.random.rand(N)
plt.scatter(x, y, s=y, marker='o')
```

```
<matplotlib.collections.PathCollection at 0x1e1909832d0>
```

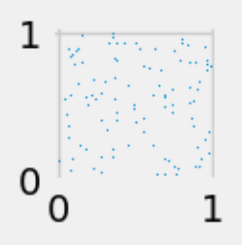


Q1.: How are the 100 random numbers represented in this graph?

2. Here is a different plot of random numbers.

```
fig=plt.figure(figsize=(1,1))
N=100
x=np.random.rand(N)
y=np.random.rand(N)
plt.scatter(x, y,s=.1,  marker='o')
```

```
<matplotlib.collections.PathCollection at 0x1e1909d1290>
```



Q2 How were random numbers used to generate the plot above?

3. The distance d from a point (x,y) to the origin (0,0) is given by the formula

$$d = \sqrt{x^2 + y^2}$$

Let's define a function dist(x,y) to compute this distance.

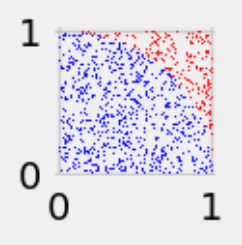
```
def dist(x,y):
    d=np.sqrt(x**2+y**2)
    return d
```

Let's use our function to check the distance from (3,4) to the origin.

Q3a What is the distance between (8,15) and the origin?

Q3b Can you find another pair of non-zero integers (a,b) whose distance to the origin is another integer c? (The integers a,b,c are called a "Pythagorean triple")

4. Let's use random numbers to create a plot of points in blue if the distance is less than or equal to 1 and in red if the distance is more than 1.



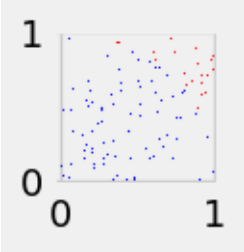
Q4 What is the shape of the points in blue?

5. The area A of a circle with radius r is $A = \pi r^2$.

Q5 What is the area of a quarter circle of radius $r = 1$?

6. Note that $\pi/4 \approx .785$. Thus, we would expect about 78.5% of the points in the plot in Step 4 to be in blue. Let's check this by counting the number of points of each color plotting a total of $N = 100$ points.

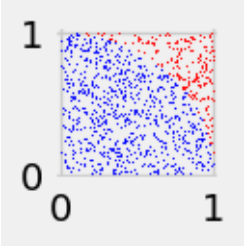
```
Percentage of Blue Points= 80.0
Estimate of Pi= 3.2
```



Q6 What would the approximation of pi be if the percentage of blue points had been 81?

7. Let's see what happens if we use $N = 1000$.

```
Percentage of Blue Points= 79.3
Estimate of Pi= 3.172
```



7. What is happening to the percentage of blue points as we increase N from 100 to 1000?

Assignment

Assignment

What do you expect to happen if we increase N to 5,000?

Check to see if you are correct.

2.5.5 Simulating Formula 1

Note

In this section, we will use an object oriented approach to create a simple animation of formula 1 cars turning a corner.

1. First we import libraries.

2. Next we create a car **class**. A class is a special programming tool similar to a function, but with additional features.

Each car has **properties** or **attributes** including:

- x position
- y position
- an image file showing what the car looks like
- a size (which can be adjusted)
- speed

Each car also has a **method** which is a type of function.

- go

```
class car:
    def __init__(self,x,y,image,size,speed):   #properties
        self.x=x  #object's position is (x,y)
        self.y=y
        self.image=image  #name of the image file
        self.size=size
        self.speed=speed

    def go(self,xamount,yamount):  #method to move the object vertically
        self.x=self.x+xamount*self.speed
        self.y = self.y+yamount*self.speed


        return
```

3. We'll define a function which positions two cars on a track.

4. We'll create two car objects and use our function to position the yellow car on the track (frame 1 of our animation).

```
c1=car(.43,.15,'car1.png',.14,0)
c2=car(.02,.01,'car2.png',.12,0)
simulate(c1,c2,0)
```

5. We'll use the go method to move the yellow car forward (frame 2 of the animation).

```
c1.speed=.8
c1.go(.2,0)
simulate(c1,c2,1)
```

6. We'll show the yellow car after it turns the corner (frame 3 of the animation).

```
c1.speed=.9
c1.size=c1.size*.9
c1.go(.2,.15)
c1.image="car1left.png"
simulate(c1,c2,2)
```

7. Now let's put the three frames together into an animation "F1.gif"

Note

Please visit the web version of this section to engage with this interactive resource.

Assignment

Assignment

Put the red car onto the track right and show it following right behind the yellow car around the turn. Then create an animation called “F1a.gif”.

2.6 Solution to Exercises

2.6.1 Beginning Python Programming

pandas

```
# 1) Create your own data file with some interesting info.
#    Read in the data into a dataframe  called df using pd.read_excel().
import pandas as pd
df=pd.read_excel("sugar.xlsx")
```

```
# 2) Display the first row of df
df.head(1)
```

```
                 Food  Grams of Sugar Recommended Daily Limit (25 grams)
0  BK Chocolate Shake             112                       4x daily limit
```

```
# 3) Abbreviate the column names.
df.columns=["food", "sugar(gm)","daily limit"]
# 4) Display the first line
df.head(1)
```

```
                 food  sugar(gm)     daily limit
0  BK Chocolate Shake         112  4x daily limit
```

```
# 5) Show how to use .loc() to get a particular entry in your dataframe.
df.loc[0,"sugar(gm)"]
```

```
112
```

for loops

```
list 1= [ 0  2  4  6  8 10]
1
3
5
7
9
11
Mission Accomplished!
```

If conditional statements

```
list2= [ 0  2  4  6  8 10 12 14 16 18 20]
6.0
7.0
8.0
9.0
10.0
```

Dataframes

```
MyCOVID2(COVID,'60637')
```

```
Zip code:  60637
population is  47454
number tested is  416617
number deaths  107
population is  47454
```

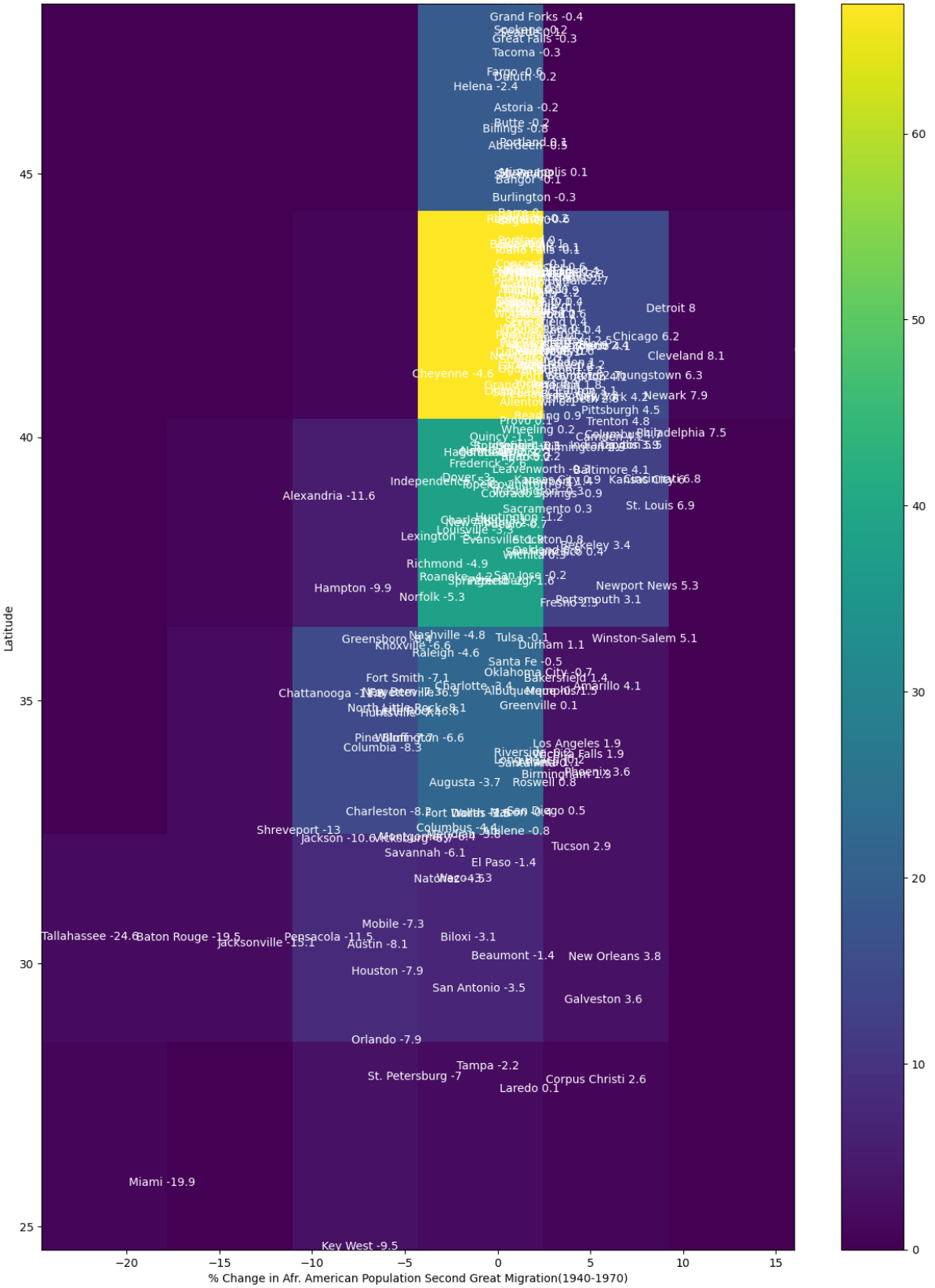
```
'Enter a different zip code if you wish.'
```

2.6.2 Glimpse of Chicago

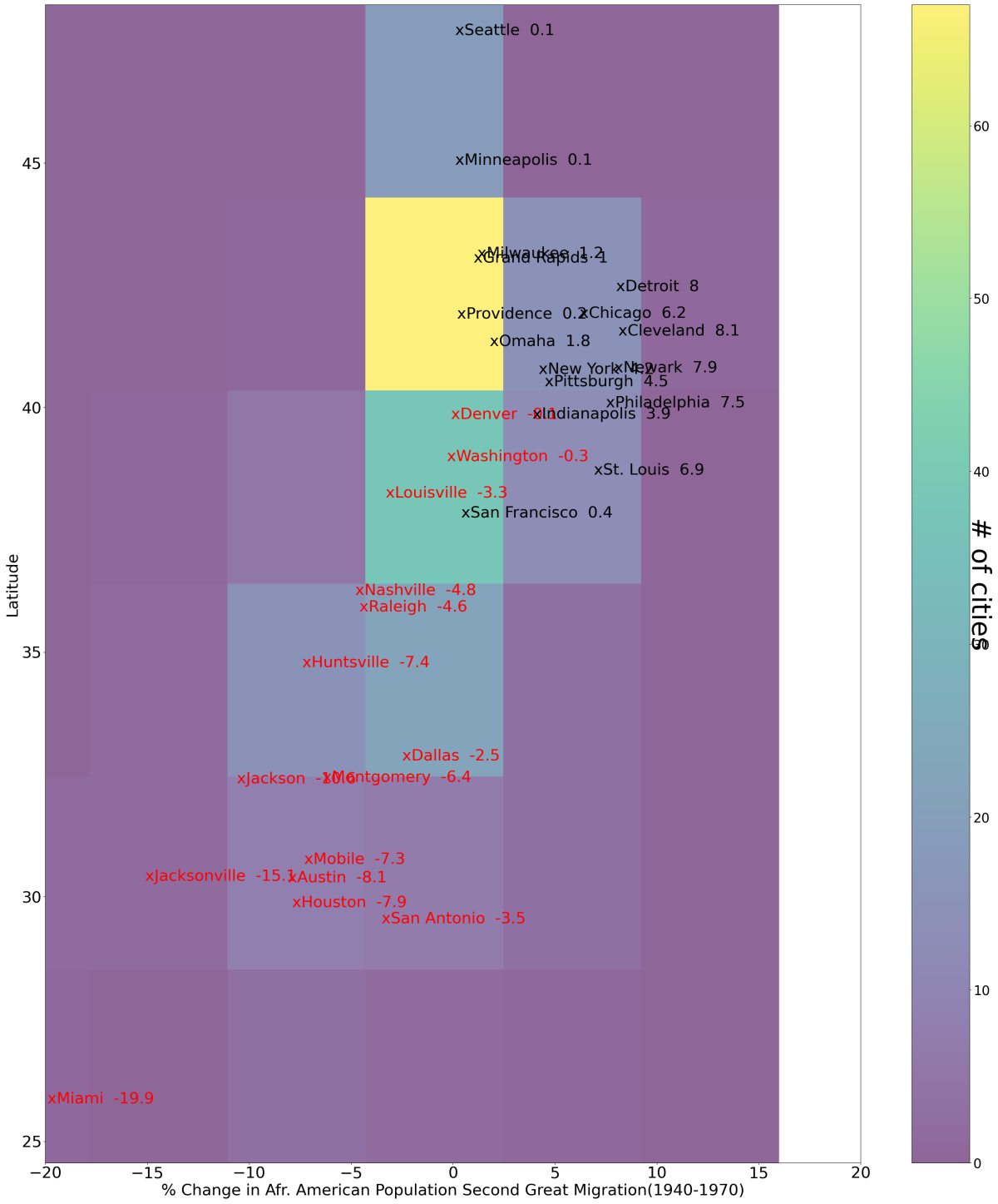
Predicting Exemplary Schools

```
|TP|= 4
|TN|= 113
|FP|= 3
|FN|= 2
Accuracy (% correct all 122 schools)= 96.0 %
Precision (% correct of those you predicted to be exemplary) = 56.99999999999999 %
Recall (% correct of schools that are exemplary) = 67.0 %
COMPETITION F1 SCORE= 61.6 %
```

Great Migration


Grand Forks -0.4
Great Falls -0.3
Tacoma -0.3
Helena -2.4
Astoria -0.2
Billings -0.8
Bangor -0.1
Burlington -0.3
Detroit 8
Chicago 6.2
Cleveland 8.1
Cheyenne -4.6
Youngstown 6.3
Newark 7.9
Pittsburgh 4.5
Trenton 4.8
Wheeling 0.2
Alexandria -11.6
St. Louis 6.9
Lexington -2
Richmond -4.9
Hampton -9.9
Norfolk -5.3
Newport News 5.3
Nashville -4.8
Tulsa -0.1
Winston-Salem 5.1
Knoxville -6.6
Durham 1.1
Raleigh -4.6
Santa Fe -0.5
Fort Smith -7.1
Amarillo 4.1
Greenville 0.1
Columbia -8.3
Los Angeles 1.9
Augusta -3.7
Roswell 0.8
Charleston -8.2
San Diego 0.5
Shreveport -13
Tucson 2.9
Savannah -6.1
El Paso -1.4
Mobile -7.3
Tallahassee -24.6
Baton Rouge -19.5
Pensacola -11.5
Austin -8.1
Biloxi -3.1
Beaumont -1.4
New Orleans 3.8
Houston -7.9
San Antonio -3.5
Galveston 3.6
Orlando -7.9
Tampa -2.2
St. Petersburg -7
Corpus Christi 2.6
Laredo 0.1
Miami -19.9
Key West -9.5
Latitude
% Change in Afr. American Population Second Great Migration(1940-1970)

```
fig,ax=plt.subplots(figsize=(35, 40))
X=df["Mig2"].astype(float)
Y=df["lat"]
heat_map= plt.hist2d(X, Y, bins=6,alpha=.6) #heat map is a 2dimensional histogram
plt.xlabel("% Change in Afr. American Population Second Great Migration(1940-1970)",
 ↪size=30)
plt.ylabel("Latitude",size=30)
plt.yticks(fontsize=30)
plt.xticks(fontsize=30)
plt.xlim(-20,20)
names=df.reset_index()
for i in names.index:   ##add the names of the cities and %change in Afr. American␣
 ↪population
    if names.Mig2[i]<0 and any(names.loc[i,'city'] in x for x in ["Austin","Chicago",
 ↪"Detroit","Cleveland","Dallas","Denver","Grand Rapids","Houston","Huntsville",
 ↪"Indianapolis","Jacksonville","Louisville","Miami","Milwaukee","Minneapolis",
 ↪"Montgomery","Mobile","Nashville","New York","New Orleans","Newark","Omaha",
 ↪"Philadelphia","Pittsburgh","Providence","Raleigh","San Antonio","San Francisco",
 ↪"Seattle","St. Louis","Washington, DC",]):
        plt.text(names.Mig2[i],names.lat[i],"x"+names.city[i]+'  '+str(names.Mig2[i]),
 ↪fontsize=30,color='red')
    else:
        if any(names.loc[i,'city'] in x for x in ["Chicago","Detroit","Cleveland",
 ↪"Dallas","Grand Rapids","Indianapolis","Louisville","Milwaukee","Minneapolis",
 ↪"Montgomery","Nashville","New York","Newark","Omaha","Philadelphia","Pittsburgh",
 ↪"Providence","Raleigh","San Francisco","Seattle","St. Louis","Washington, DC",]):
            plt.text(names.Mig2[i],names.lat[i],"x"+names.city[i]+'  '+str(names.
 ↪Mig2[i]),fontsize=30,color='black')

cbar = plt.colorbar()
cbar.set_label('# of cities', rotation=270,size=50)
cbar.ax.tick_params(labelsize=25)
fig.savefig("Migration2Simplified.png")
```



The percentage changes are smaller for the 2nd great migration when compared to the first. For example, Chicago dropped from 24.6 to 6.2.

Chicago Hardship Index

```
import pandas as pd
raw_hardship=pd.read_excel('HIHOM20142017.xlsx')
dfHI14=raw_hardship[["Community","population(K)","HI14","UNEMP14","NOHS14","DEP14",
```

(continues on next page)

(continued from previous page)

```
 ↪"HOUS14","POV14","INC14","HOM14","LAT","LON"]]
dfHI17=raw_hardship[["Community","population(K)","HI17","UNEMP17","NOHS17","DEP17",
 ↪"HOUS17","POV17","INC17","HOM17","LAT","LON"]]
hom_df = dfHI17
```

```
import matplotlib.pyplot as plt
fig=plt.figure(figsize=(25,20))

for i in hom_df.index:

    if hom_df.loc[i,"HOM17"]==0:
        plt.scatter(hom_df.loc[i,'LON'], hom_df.loc[i,'LAT'],s=30,color='b', alpha=0.
 ↪2)
        plt.gca().text(hom_df.loc[i,'LON']-.001, hom_df.loc[i,'LAT']-.001,hom_df.
 ↪loc[i,'HOM17'],color='b', alpha=1,size=20)
        if hom_df.loc[i,"Community"] in ["Loop"]:
            plt.gca().text(hom_df.loc[i,'LON']-.005, hom_df.loc[i,'LAT']+.005, hom_df.
 ↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='blue', size=30)
        elif hom_df.loc[i,"Community"] in ["Hyde Park","Near West Side","Kenwood",
 ↪"Near North Side","West Town"]:
            plt.gca().text(hom_df.loc[i,'LON']+.003, hom_df.loc[i,'LAT']-.005, hom_df.
 ↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='blue', size=20)
        elif hom_df.loc[i,"Community"] in ["Lincoln Park","Uptown","Edgewater",
 ↪"Rogers Park","Logan Square","Avondale"]:
            plt.gca().text(hom_df.loc[i,'LON']-.015, hom_df.loc[i,'LAT']-.0075, hom_
 ↪df.loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='blue', size=20)
        elif hom_df.loc[i,"Community"] in ["Lincoln Square","Irving Park","Portage␣
 ↪Park","Jefferson Park"]:
            plt.gca().text(hom_df.loc[i,'LON']-.025, hom_df.loc[i,'LAT']-.0075, hom_
 ↪df.loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='blue', size=20)
        elif hom_df.loc[i,"Community"] in ["Dunning","O'Hare","Edison Park","Norwood␣
 ↪Park"]:
            plt.gca().text(hom_df.loc[i,'LON']-.015, hom_df.loc[i,'LAT']+.005, hom_df.
 ↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='blue', size=20)

        elif hom_df.loc[i,"Community"] in ["Garfield Ridge","Clearning","Ashburn",
 ↪"Morgan Park"]:
            plt.gca().text(hom_df.loc[i,'LON']-.015, hom_df.loc[i,'LAT']+.0015, hom_
 ↪df.loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='blue', size=20)
        elif hom_df.loc[i,"Community"] in ["Hegewisch"]:
            plt.gca().text(hom_df.loc[i,'LON']-.01, hom_df.loc[i,'LAT']-.0075, hom_df.
 ↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='blue', size=20)
        elif hom_df.loc[i,"Community"] in ["Forest Glen"]:
            plt.gca().text(hom_df.loc[i,'LON']-.025, hom_df.loc[i,'LAT']+.005, hom_df.
 ↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='blue', size=20)
        elif hom_df.loc[i,"Community"] in ["North Center"]:
            plt.gca().text(hom_df.loc[i,'LON']-.02, hom_df.loc[i,'LAT']+.005, hom_df.
 ↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='blue', size=20)
        elif hom_df.loc[i,"Community"] in ["Oakland"]:
            plt.gca().text(hom_df.loc[i,'LON']-.01, hom_df.loc[i,'LAT']+.005, hom_df.
 ↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='blue', size=20)
        elif hom_df.loc[i,"Community"] in ["Mount Greenwood"]:
            plt.gca().text(hom_df.loc[i,'LON']-.025, hom_df.loc[i,'LAT']+.005, hom_df.
 ↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='blue', size=20)
        elif hom_df.loc[i,"Community"] in ["Lakeview"]:
            plt.gca().text(hom_df.loc[i,'LON']-.015, hom_df.loc[i,'LAT']+.005, hom_df.
```

(continues on next page)

(continued from previous page)

```
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='blue', size=20)
        elif hom_df.loc[i,"Community"] in ["Near South Side"]:
            plt.gca().text(hom_df.loc[i,'LON']-.015, hom_df.loc[i,'LAT']+.005, hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='blue', size=20)

        elif hom_df.loc[i,"Community"] in ["Edison Park"]:
            plt.gca().text(hom_df.loc[i,'LON']-.015, hom_df.loc[i,'LAT']+.005, hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='green', size=20)
        elif hom_df.loc[i,"Community"] in ["Beverly"]:
            plt.gca().text(hom_df.loc[i,'LON']-.02, hom_df.loc[i,'LAT']+.0045, hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='blue', size=20)


        else:
            plt.gca().text(hom_df.loc[i,'LON']-.02,hom_df.loc[i,'LAT']+.005,hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='blue', size=20)

    else:
        p=hom_df.loc[i,'population(K)']
        plt.scatter(hom_df.loc[i,'LON'], hom_df.loc[i,'LAT'],s=30+1000*hom_df.loc[i,
↪'HOM17']/p,color='k', alpha=0.2)
        plt.gca().text(hom_df.loc[i,'LON']-.0025, hom_df.loc[i,'LAT']-.0015,
↪str(round(hom_df.loc[i,'HOM17']/p,2)),color='k', alpha=1,size=20)
        if hom_df.loc[i,"Community"] in ["Austin"]:
            plt.gca().text(hom_df.loc[i,'LON']-.02,hom_df.loc[i,'LAT']+.013, hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Belmont Craigin"]:
            plt.gca().text(hom_df.loc[i,'LON']-.02,hom_df.loc[i,'LAT']+.0078, hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Chicago Lawn"]:
            plt.gca().text(hom_df.loc[i,'LON']-.015,hom_df.loc[i,'LAT']-.009, hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Pullman"]:
            plt.gca().text(hom_df.loc[i,'LON']-.015,hom_df.loc[i,'LAT']+.002,hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Roseland"]:
            plt.gca().text(hom_df.loc[i,'LON']-.015,hom_df.loc[i,'LAT']+.007,hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["West Garfield Park"]:
            plt.gca().text(hom_df.loc[i,'LON']-.04,hom_df.loc[i,'LAT']+.005,hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Oakland","Armour Square"]:
            plt.gca().text(hom_df.loc[i,'LON']-.001,hom_df.loc[i,'LAT']+.002,hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Bridgeport"]:
            plt.gca().text(hom_df.loc[i,'LON']-.029,hom_df.loc[i,'LAT']+.003,hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["South Shore"]:
            plt.gca().text(hom_df.loc[i,'LON']-.015,hom_df.loc[i,'LAT']+.003,hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["West Pullman"]:
            plt.gca().text(hom_df.loc[i,'LON']-.025,hom_df.loc[i,'LAT']+.003,hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Washington Park"]:
            plt.gca().text(hom_df.loc[i,'LON']-.03,hom_df.loc[i,'LAT']+.003,hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
```

(continues on next page)

(continued from previous page)

```
        elif hom_df.loc[i,"Community"] in ["Washington Heights"]:
            plt.gca().text(hom_df.loc[i,'LON']-.05,hom_df.loc[i,'LAT']+.005,hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["West Ridge","Albany Park"]:
            plt.gca().text(hom_df.loc[i,'LON']-.02,hom_df.loc[i,'LAT']+.005,hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["North Park"]:
            plt.gca().text(hom_df.loc[i,'LON']-.01,hom_df.loc[i,'LAT']+.005,hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Lincoln Square"]:
            plt.gca().text(hom_df.loc[i,'LON']-.019,hom_df.loc[i,'LAT']+.002,hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Hermosa"]:
            plt.gca().text(hom_df.loc[i,'LON']-.02,hom_df.loc[i,'LAT']+.003,hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Montclare"]:
            plt.gca().text(hom_df.loc[i,'LON']-.02,hom_df.loc[i,'LAT']-.007, hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["West Town"]:
            plt.gca().text(hom_df.loc[i,'LON']-.02,hom_df.loc[i,'LAT']+.005, hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Near West Side"]:
            plt.gca().text(hom_df.loc[i,'LON']-.01,hom_df.loc[i,'LAT']+.005, hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Lower West Side"]:
            plt.gca().text(hom_df.loc[i,'LON']-.03,hom_df.loc[i,'LAT']+.005,hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["North Lawndale"]:
            plt.gca().text(hom_df.loc[i,'LON']-.015,hom_df.loc[i,'LAT']+.006,hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Douglas"]:
            plt.gca().text(hom_df.loc[i,'LON']-.009,hom_df.loc[i,'LAT']+.004,hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Archer Heights"]:
            plt.gca().text(hom_df.loc[i,'LON']-.03,hom_df.loc[i,'LAT']+.005, hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["New City"]:
            plt.gca().text(hom_df.loc[i,'LON']-.015,hom_df.loc[i,'LAT']+.01, hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["South Chicago"]:
            plt.gca().text(hom_df.loc[i,'LON']-.015,hom_df.loc[i,'LAT']+.004, hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["West Lawn"]:
            plt.gca().text(hom_df.loc[i,'LON']-.025,hom_df.loc[i,'LAT']+.005, hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Kenwood"]:
            plt.gca().text(hom_df.loc[i,'LON']+.002,hom_df.loc[i,'LAT'], hom_df.loc[i,
↪'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Grand Boulevard"]:
            plt.gca().text(hom_df.loc[i,'LON']-.001,hom_df.loc[i,'LAT']+.005,hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Fuller Park"]:
            plt.gca().text(hom_df.loc[i,'LON']-.03,hom_df.loc[i,'LAT']+.005,hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Hyde Park"]:
            plt.gca().text(hom_df.loc[i,'LON']-.001,hom_df.loc[i,'LAT']+.003,hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
```

(continues on next page)

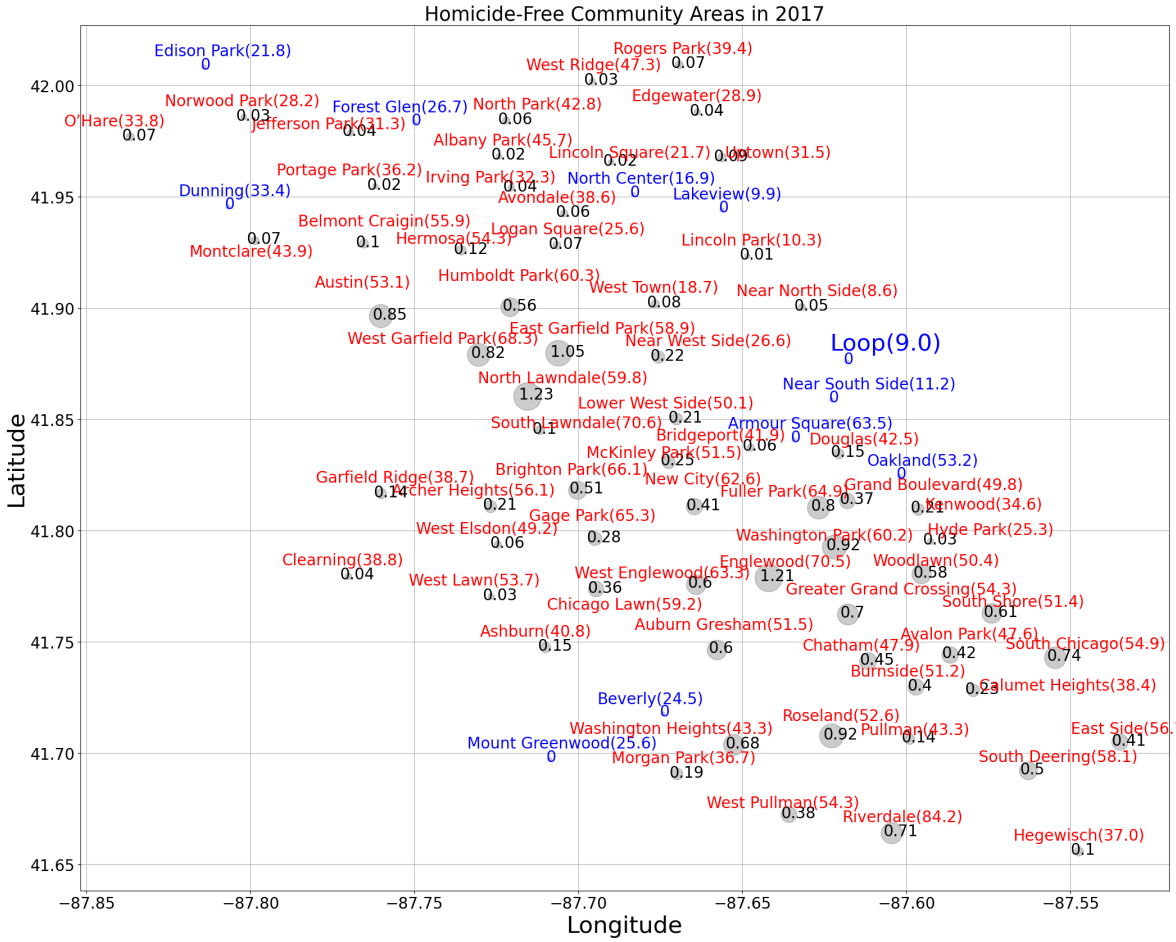
(continued from previous page)

```
        elif hom_df.loc[i,"Community"] in ["Calumet Heights"]:
            plt.gca().text(hom_df.loc[i,'LON']+.002,hom_df.loc[i,'LAT'],hom_df.loc[i,
↪'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Jefferson Park"]:
            plt.gca().text(hom_df.loc[i,'LON']-.03,hom_df.loc[i,'LAT']+.0015,hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Uptown"]:
            plt.gca().text(hom_df.loc[i,'LON']+.001,hom_df.loc[i,'LAT'],hom_df.loc[i,
↪'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["West Englewood"]:
            plt.gca().text(hom_df.loc[i,'LON']-.037,hom_df.loc[i,'LAT']+.003,hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Portage Park"]:
            plt.gca().text(hom_df.loc[i,'LON']-.03,hom_df.loc[i,'LAT']+.005,hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Irving Park"]:
            plt.gca().text(hom_df.loc[i,'LON']-.026, hom_df.loc[i,'LAT']+.0025, hom_
↪df.loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Humboldt Park"]:
            plt.gca().text(hom_df.loc[i,'LON']-.022,hom_df.loc[i,'LAT']+.012,hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Brighton Park"]:
            plt.gca().text(hom_df.loc[i,'LON']-.025,hom_df.loc[i,'LAT']+.007, hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["South Lawndale"]:
            plt.gca().text(hom_df.loc[i,'LON']-.015,hom_df.loc[i,'LAT']+.001, hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["McKinley Park"]:
            plt.gca().text(hom_df.loc[i,'LON']-.025,hom_df.loc[i,'LAT']+.002, hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["West Elsdon"]:
            plt.gca().text(hom_df.loc[i,'LON']-.025,hom_df.loc[i,'LAT']+.005, hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Gage Park"]:
            plt.gca().text(hom_df.loc[i,'LON']-.02,hom_df.loc[i,'LAT']+.008, hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Woodlawn"]:
            plt.gca().text(hom_df.loc[i,'LON']-.015,hom_df.loc[i,'LAT']+.004, hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Englewood"]:
            plt.gca().text(hom_df.loc[i,'LON']-.015,hom_df.loc[i,'LAT']+.005, hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Greater Grand Crossing"]:
            plt.gca().text(hom_df.loc[i,'LON']-.019,hom_df.loc[i,'LAT']+.009, hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Auburn Gresham"]:
            plt.gca().text(hom_df.loc[i,'LON']-.025,hom_df.loc[i,'LAT']+.009, hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Avalon Park"]:
            plt.gca().text(hom_df.loc[i,'LON']-.015,hom_df.loc[i,'LAT']+.007,hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Burnside"]:
            plt.gca().text(hom_df.loc[i,'LON']-.02,hom_df.loc[i,'LAT']+.005,hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["East Side","South Deering"]:
            plt.gca().text(hom_df.loc[i,'LON']-.015,hom_df.loc[i,'LAT']+.004, hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
```

(continues on next page)

(continued from previous page)

```
        elif hom_df.loc[i,"Community"] in ["Riverdale"]:
            plt.gca().text(hom_df.loc[i,'LON']-.015,hom_df.loc[i,'LAT']+.005, hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["Norwood Park"]:
            plt.gca().text(hom_df.loc[i,'LON']-.024,hom_df.loc[i,'LAT']+.005, hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)
        elif hom_df.loc[i,"Community"] in ["East Garfield Park"]:
            plt.gca().text(hom_df.loc[i,'LON']-.015,hom_df.loc[i,'LAT']+.009,hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)

        else:
            plt.gca().text(hom_df.loc[i,'LON']-.02,hom_df.loc[i,'LAT']+.005,hom_df.
↪loc[i,'Community']+'('+str(hom_df.loc[i,'HI17'])+')',color='red', size=20)

#plt.gca().set_facecolor('lightgray')
plt.gca().grid()
plt.yticks(fontsize=20)
plt.xticks(fontsize=20)
#title
plt.title('Homicide-Free Community Areas in 2017',size=25)
plt.xlabel("Longitude",size=30)
plt.ylabel("Latitude",size=30)
fig.savefig("Fig5revised.png")
#show the plot
plt.show()
```



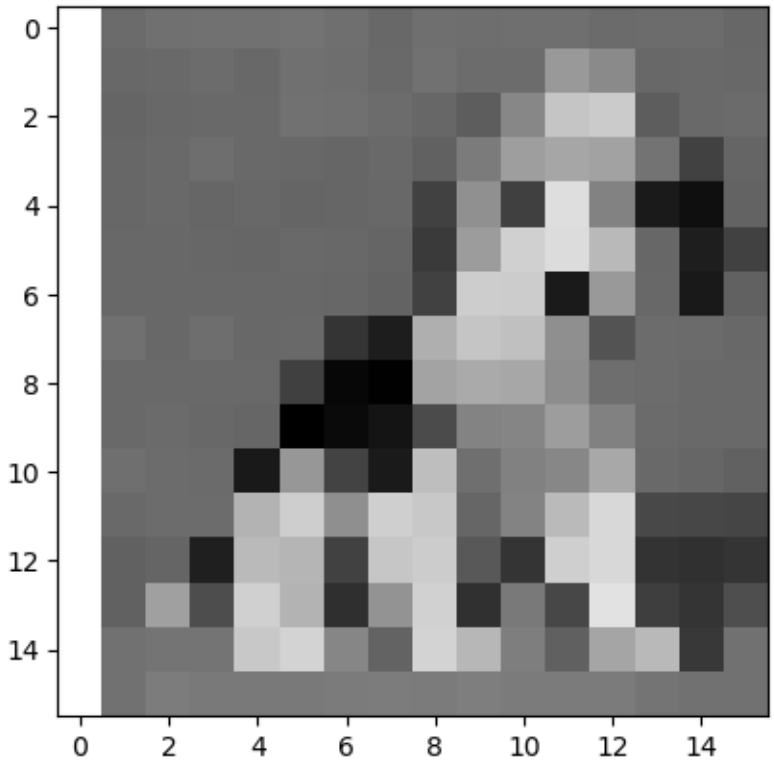
2.6.3 Arts in STEM(STEAM)

Pixel Images

1a) By increasing the number of pixels, we can get a better reproduction of the original. Create a 16x16 pixel image.

```
makepixelimage("images", 16)
```

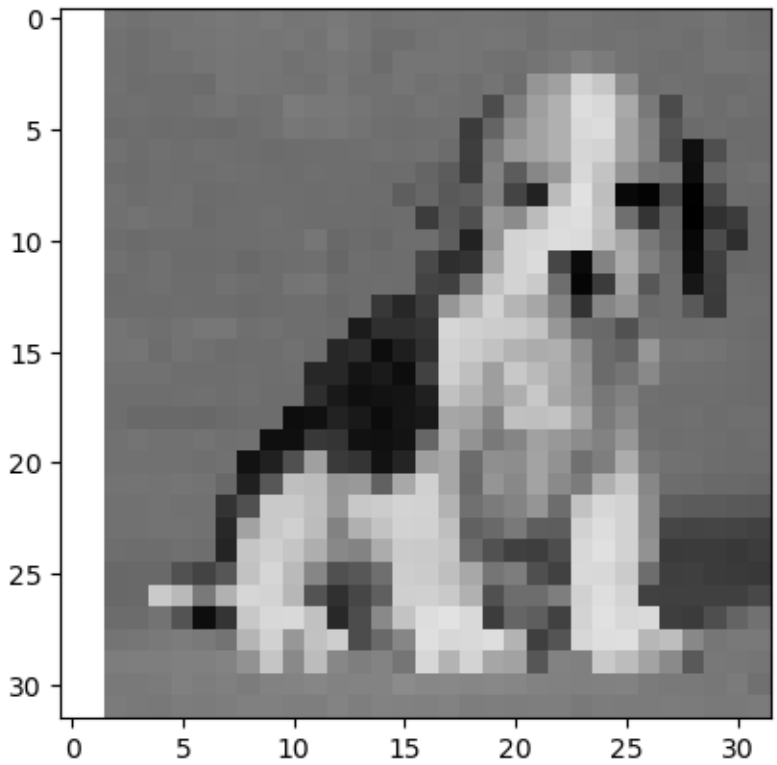
```
<matplotlib.image.AxesImage at 0x1c256419dd0>
```



1b) Create a 32x32 pixel image.

```
makepixelimage("images", 32)
```

```
<matplotlib.image.AxesImage at 0x1c256d7f8d0>
```

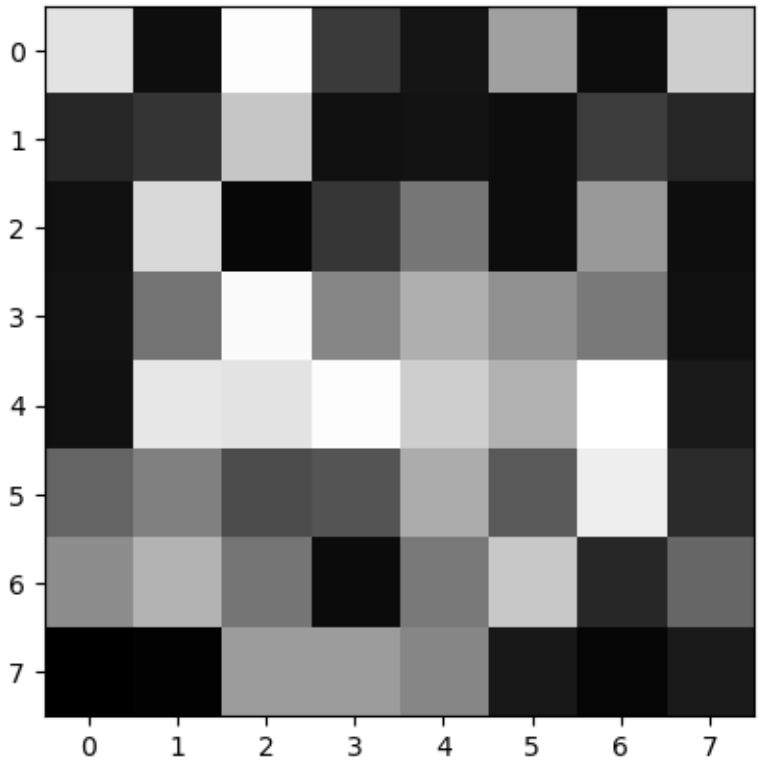


2) Upload a different image into a folder called “my_images”. Then use the makepixelimage(,) function to create an 8x8, 16x16 and 32x32 pixel image and see if the others in the class can guess your original image

Many answers possible.

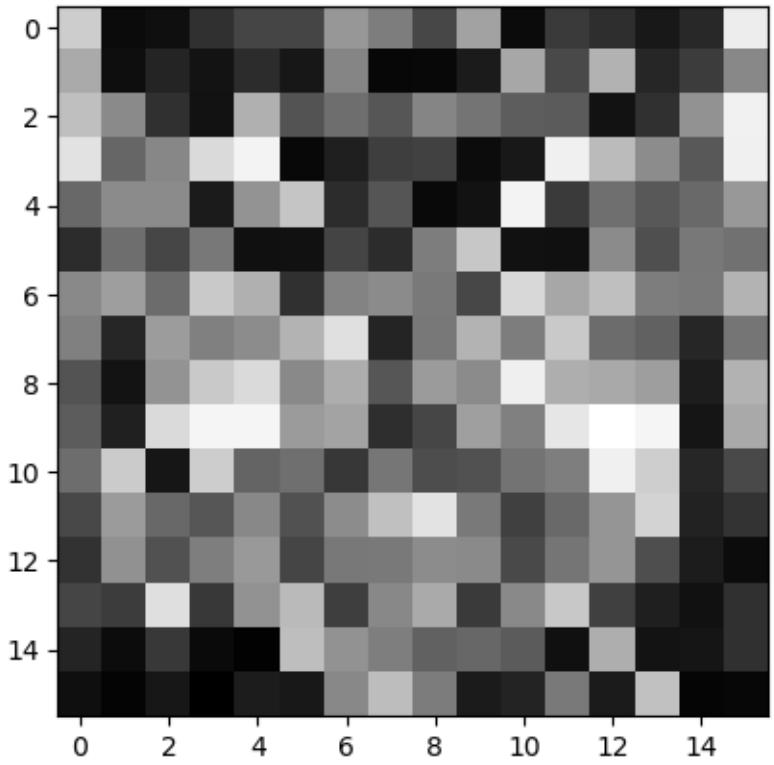
```
makepixelimage("my_images", 8)
```

```
<matplotlib.image.AxesImage at 0x1c256e078d0>
```



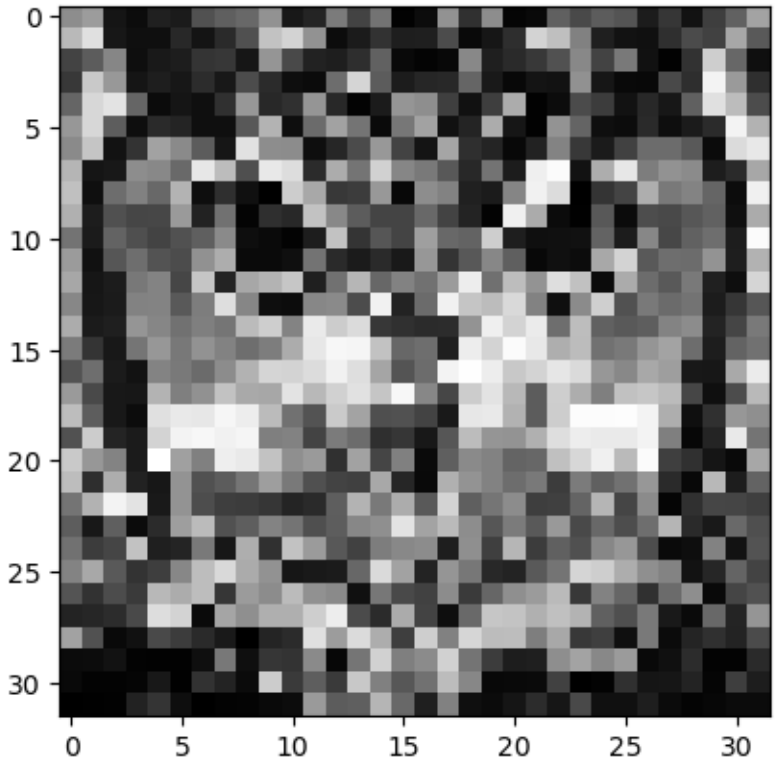

```
makepixelimage("my_images", 16)
```

```
<matplotlib.image.AxesImage at 0x1c2562c4550>
```




```
makepixelimage("my_images", 32)
```

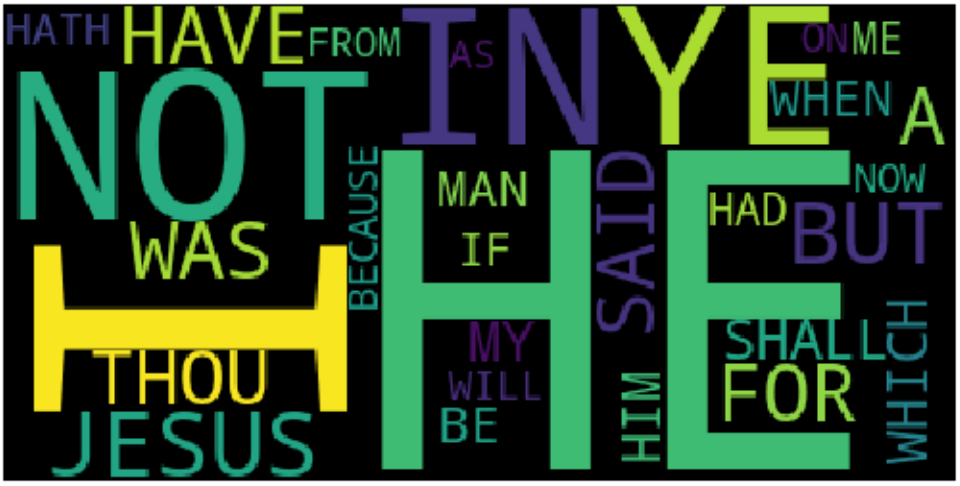
```
<matplotlib.image.AxesImage at 0x1c257efa5d0>
```



WORD CLOUDS Many Possible Answers

```
with open('john.txt','r') as file:  #read in the text file
    gospeljohn = file.readlines()
```

Word Cloud of Gospel of John with 30 words



Name that Tune! many possible answers

Note

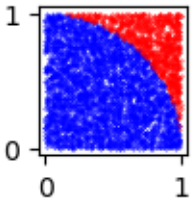
Please visit the web version of this section to engage with this interactive resource.

```
saintsaens = [so, so , so, so, la, la,la,la,so, so , do1, do1, so, so,fa,fa,mi,mi,fa,
↪so,la,la,fa,fa,re,re,re,re,so,so,so,so]
play(saintsaens)
```

```
<IPython.lib.display.Audio object>
```

Random Numbers and Pi

```
Percentage of Blue Points= 78.24
Estimate of Pi= 3.1296
```



Simulating Formula 1

Note

Please visit the web version of this section to engage with this interactive resource.

2.7 JNB LAB: After-School Program Demo

Note

One way to wrap-up an after-school program is to have the students do a combined presentation showing several things they learned throughout the program. It is a good idea to practice the presentations at least once or twice in advance.

Today our class will demonstrate several things which can be done with Jupyter Notebooks.

First we load standard libraries for analyzing and plotting data.

```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
```

2.7.1 DEMO 1: City of Chicago Budget

We will make a piechart of the City of Chicago budget.

1) First let's look at the 2022 City of Chicago revenue details.

Summary A
COMPARATIVE STATEMENT OF CORPORATE FUND REVENUES BY MAJOR SOURCES FOR THE YEARS 2020, 2021 AND 2022

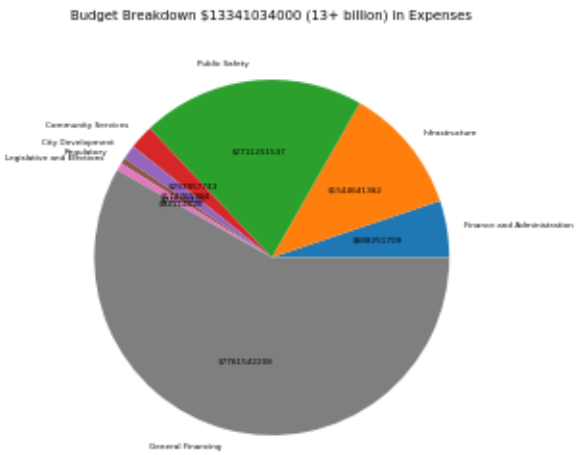
Sources	Revised 2020	Published 2021	Revised 2021	Estimated 2022
Local Tax				
Municipal Public Utility Tax	$416,114,000	$390,803,000	$390,803,000	$388,902,000
Chicago Sales Tax / Home Rule Retailers' Occupation Tax	74,043,000	63,639,000	63,639,000	73,055,000
Transaction Taxes	508,851,000	478,129,000	478,129,000	582,313,000
Transportation Taxes	383,586,000	308,693,000	308,693,000	336,087,000
Recreation Taxes	273,349,000	223,911,000	223,911,000	274,350,000
Business Taxes	134,400,000	66,122,000	66,122,000	112,744,000
Total - Local Tax	**$1,790,343,000**	**$1,531,297,000**	**$1,531,297,000**	**$1,767,451,000**
Proceeds and Transfers In				
Proceeds and Transfers In	$642,510,000	$612,179,000	$612,179,000	$1,048,933,000
Total - Proceeds and Transfers In	**$642,510,000**	**$612,179,000**	**$612,179,000**	**$1,048,933,000**
Intergovernmental Revenue				
State Income Tax	$269,780,000	$262,279,000	$262,279,000	$366,600,000
Personal Property Replacement Tax	157,223,000	147,000,000	147,000,000	163,683,000
Municipal Auto Rental Tax	4,150,000	3,332,000	3,332,000	3,960,000
Reimbursements for City Services	2,000,000	2,000,000	2,000,000	2,000,000
Total - Intergovernmental Revenue	**$433,153,000**	**$414,611,000**	**$414,611,000**	**$536,243,000**
Local Non-Tax Revenue				
Licenses, Permits, and Certificates	$128,300,000	$119,200,000	$119,200,000	$119,600,000
Fines, Forfeitures and Penalties	342,700,000	381,500,000	381,500,000	369,700,000
Charges for Services	460,200,000	277,900,000	277,900,000	340,400,000
Municipal Parking	7,600,000	7,600,000	7,600,000	7,600,000
Leases, Rentals and Sales	33,700,000	39,300,000	39,300,000	29,300,000
Interest Income	3,500,000	6,500,000	6,500,000	6,500,000
Internal Service Earnings	361,463,000	410,732,000	410,732,000	485,773,000
Other Revenue	134,745,000	125,820,000	125,820,000	121,097,000
Total - Local Non-Tax Revenue	**$1,472,208,000**	**$1,368,552,000**	**$1,368,552,000**	**$1,479,970,000**
Total - All Sources	**$4,338,214,000**	**$3,926,639,000**	**$3,926,639,000**	**$4,832,597,000**
Net Current Assets at January 1	81,000,000	111,000,000	111,000,000	51,400,000
Net Total - All Sources	**$4,419,214,000**	**$4,037,639,000**	**$4,037,639,000**	**$4,883,997,000**

2) Next let's read an Excel sheet with the summary of the 2023 budget.

```
budget=pd.read_excel('ChicagoBudget.xlsx')
budget
```

```
                     EXPENSE  2023 BUDGET
0  Finance and Administration    688251735
1              Infrastructure   1544641397
2               Public Safety   2711251614
3          Community Services    293957760
4            City Development    174766005
5                  Regulatory     74509317
6   Legislative and Elections     92114035
7           General Financing   7761542137
```

3) A pie chart will show us the proportions.



Exercise

Exercise

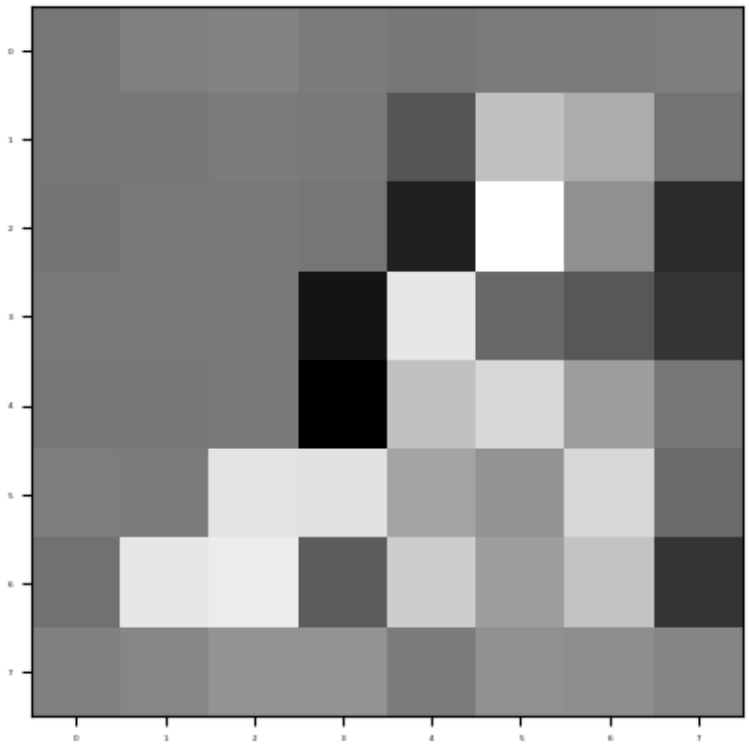
Enlarge the size of the piechart so it is easier to read. Make sure you enlarge the fontsize as well as the chart size.

2.7.2 Demo 2 Pixel Images

We can increase the resolution of images by increasing the number of pixels.

```
makepixelimage("images", 8)
```

```
<matplotlib.image.AxesImage at 0x18fccc85110>
```



Exercise

Increase the image resolution to 16x16 and then 32x32 so the image will become much clearer.

2.7.3 DEMO 3: Track NFL Player Positions

We will plot the movement of two players step by step in a given play Dataset: NFL_play.xlsx

1) Import special libraries.

2) Read the Player Tracking Data

```
track_play=pd.read_excel('NFL_play.xlsx')
track_play.head(22)
```

```
    Unnamed: 0     game_play  game_key  play_id  nfl_player_id  \
0            0  58580_001136     58580     1136          44830
1            1  58580_001136     58580     1136          42355
2            2  58580_001136     58580     1136          43330
3            3  58580_001136     58580     1136          52425
4            4  58580_001136     58580     1136          43293
5            5  58580_001136     58580     1136          40031
6            6  58580_001136     58580     1136          41242
7            7  58580_001136     58580     1136          52938
8            8  58580_001136     58580     1136          42347
9            9  58580_001136     58580     1136          46135
10          10  58580_001136     58580     1136          43424
11          11  58580_001136     58580     1136          43351
```

(continues on next page)

(continued from previous page)

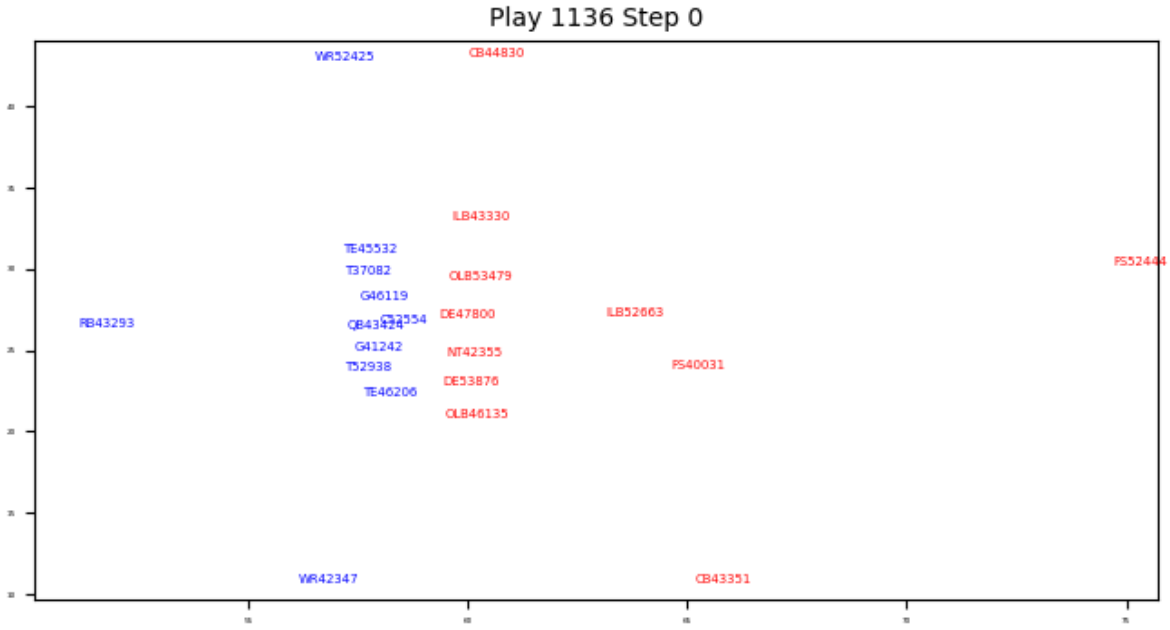
```
12          12  58580_001136     58580     1136          45532
13          13  58580_001136     58580     1136          46119
14          14  58580_001136     58580     1136          37082
15          15  58580_001136     58580     1136          53876
16          16  58580_001136     58580     1136          53479
17          17  58580_001136     58580     1136          52663
18          18  58580_001136     58580     1136          46206
19          19  58580_001136     58580     1136          52444
20          20  58580_001136     58580     1136          47800
21          21  58580_001136     58580     1136          52554

                   datetime  step  team position  jersey_number  x_position  \
0   2021-10-10T21:08:20.900Z  -108  away       CB             22       61.59
1   2021-10-10T21:08:20.900Z  -108  away       NT             75       59.63
2   2021-10-10T21:08:20.900Z  -108  away      ILB             55       60.67
3   2021-10-10T21:08:20.900Z  -108  home       WR             88       56.59
4   2021-10-10T21:08:20.900Z  -108  home       RB             21       51.11
5   2021-10-10T21:08:20.900Z  -108  away       FS             23       70.53
6   2021-10-10T21:08:20.900Z  -108  home        G             70       57.33
7   2021-10-10T21:08:20.900Z  -108  home        T             78       57.27
8   2021-10-10T21:08:20.900Z  -108  home       WR             19       56.23
9   2021-10-10T21:08:20.900Z  -108  away      OLB             59       59.90
10  2021-10-10T21:08:20.900Z  -108  home       QB              4       57.31
11  2021-10-10T21:08:20.900Z  -108  away       CB             24       64.39
12  2021-10-10T21:08:20.900Z  -108  home       TE             89       57.16
13  2021-10-10T21:08:20.900Z  -108  home        G             52       57.36
14  2021-10-10T21:08:20.900Z  -108  home        T             77       57.16
15  2021-10-10T21:08:20.900Z  -108  away       DE             91       59.65
16  2021-10-10T21:08:20.900Z  -108  away      OLB             51       59.47
17  2021-10-10T21:08:20.900Z  -108  away      ILB             48       63.25
18  2021-10-10T21:08:20.900Z  -108  home       TE             86       57.37
19  2021-10-10T21:08:20.900Z  -108  away       FS             29       72.19
20  2021-10-10T21:08:20.900Z  -108  away       DE             97       59.48
21  2021-10-10T21:08:20.900Z  -108  home        C             63       58.18

    y_position  speed  distance  direction  orientation  acceleration    sa
0        42.60   1.11      0.11     320.33       263.93          0.71 -0.64
1        24.33   0.10      0.01       7.98       227.03          0.41  0.27
2        30.89   3.19      0.32     334.89       303.31          1.95 -1.73
3        42.86   0.13      0.01     158.78        98.31          0.32  0.02
4        26.42   0.14      0.01     144.58        78.52          0.52  0.51
5        22.03   0.32      0.03     285.68       287.44          0.28  0.27
6        24.80   0.03      0.01     328.04        57.38          0.07  0.07
7        23.47   0.19      0.02     356.50        87.29          0.10 -0.10
8        10.68   0.07      0.01     132.91       123.39          0.19 -0.14
9        21.14   1.58      0.16     218.10       278.39          0.48  0.22
10       26.27   0.07      0.01     207.25        93.25          0.11  0.07
11       10.89   0.80      0.08     130.49       309.04          0.44 -0.44
12       30.99   0.07      0.01      39.61       100.73          0.51 -0.14
13       28.02   0.08      0.01     127.21       241.42          0.05  0.04
14       29.54   0.12      0.01     145.27        85.16          0.07  0.01
15       22.71   0.23      0.02     268.69       271.27          0.23 -0.23
16       29.52   0.24      0.02     296.78       257.94          1.12  0.26
17       27.50   0.51      0.05     183.62       253.71          0.31  0.31
18       22.12   0.37      0.04     127.85        63.63          0.69  0.62
19       31.46   0.61      0.06      11.77       247.69          0.63 -0.33
20       26.81   0.23      0.01     346.84       247.16          1.29  0.90
```

(continues on next page)

(continued from previous page)

```
21        26.52    0.16       0.02      357.62        102.55           0.60   0.58
```

3) Plot the positions of the players at step -108 (before the snap) of play 1136.



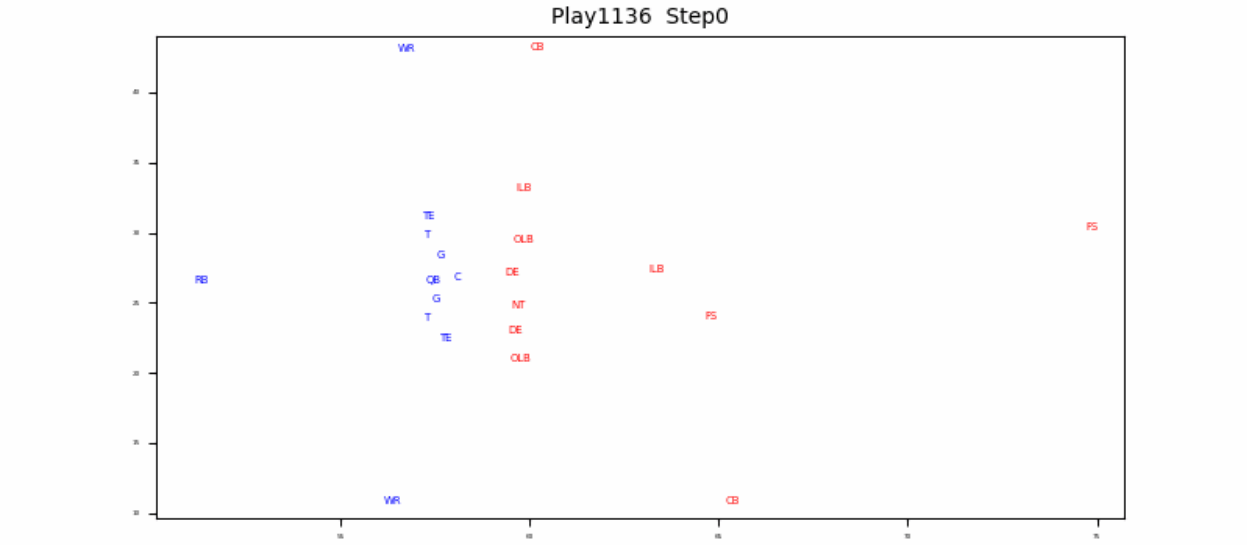
4) Let's define a function which creates a snapshot of the position of the players at any step of a given play.

5. Let's use this function to create snapshots of player positions for the first 50 steps after the snap of play 1136.

6. Let's combine these snapshots into an animation. Can you figure out what happened on this play?

Note

Please visit the web version of this section to engage with this interactive resource.



Exercise

Exercise

Create a video which isolates the movement of just two players: the wide receiver (52425) and cornerback (44830) at the top of the screen.

2.7.4 Demo 4 Word Clouds

Let's make a word cloud Christmas card using the song "Twelve Days of Christmas."

```
import wordcloud
```

```
#Open the text file with the words to be plotted.
with open('twelvedays.txt','r') as file:
    carol = file.readlines()

#make the wordcloud
carol = calculate_frequencies(carol)
plt.imshow(carol, interpolation = 'nearest')
plt.axis('off')
plt.savefig('card.png', bbox_inches='tight')
```

Exercise

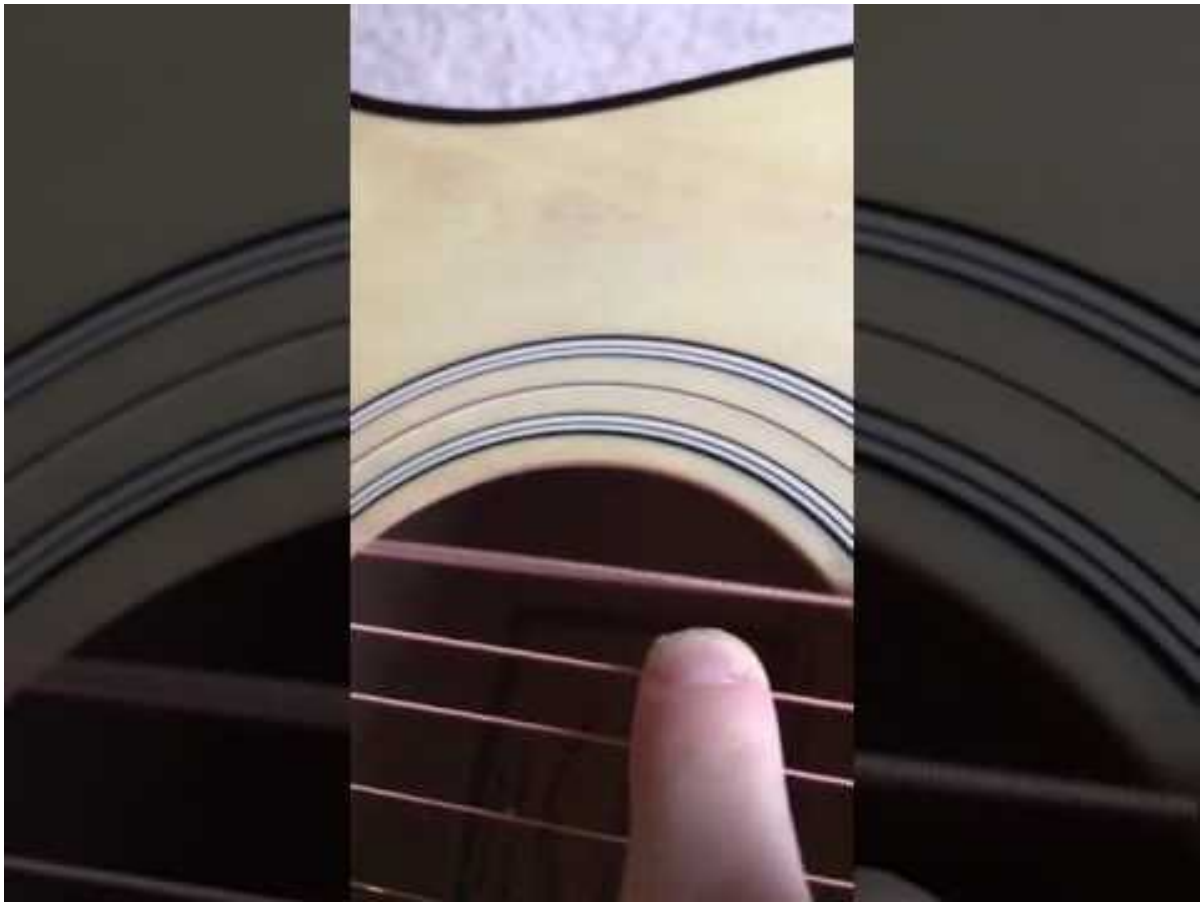
Exercise

Add the words “Merry Christmas!” in red onto the middle of the wordcloud.

2.7.5 Demo 5 Name that Tune

1) Musical sound waves are created by rapid vibrations caused by musical instruments.

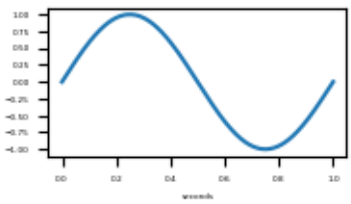
```
from IPython.display import YouTubeVideo
YouTubeVideo('tVYQRC1-D54')
```



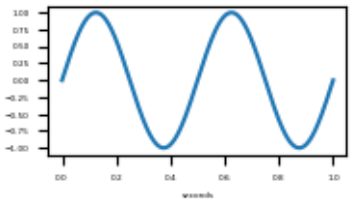
2) Sound waves are represented mathematically by sine waves with different frequencies.

```
def sinewave(frequency):
    #-----------CREATE THE SOUND WAVE-------------------
    sampling_rate=44100  #how many times we take a measurement each second
    t = np.linspace(0,1,sampling_rate)  # take 44100 samples in 1 second;
    sound_wave=np.sin(frequency* 2*np.pi* t)  # mathematical definition of a sine wave
    #----------PLOT THE SOUND WAVE----------------------
    import matplotlib.pyplot as plt
    fig=plt.figure(figsize=(2,1))
    plt.plot(t,sound_wave)
    plt.xlabel("seconds")
    return
```

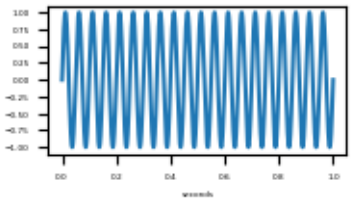
```
sinewave(1)  #frequency=1 and 1 cycle per second
```



```
sinewave(2)  #frequency=2 and 2 cycles per second
```



```
sinewave(20) #frequency=20 and 20 cycles per second
```



3) A computer can create a musical tone based on a given frequency.

```
def play(freq):
    import numpy as np
    from IPython.display import Audio  #library used to create sounds
    sampling_rate = 44100 # <- rate of sampling
    t = np.linspace(0, 2,  sampling_rate) # <- setup time values
    sound_wave = np.sin(2 * np.pi * freq * t) # <- sine function formula
    return Audio(sound_wave, rate=sampling_rate, autoplay=True) # play the generated␣
↪sound
```

Note

Please visit the web version of this section to engage with this interactive resource.

```
play(220) # play a sound at 220 hz
```

```
<IPython.lib.display.Audio object>
```

4) A musical scale is a sequence of frequencies.

```
from IPython.display import Audio
rest=0
do=220
re=9/8*220
mi=5/4*220
fa=4/3*220
so=3/2*220
la=5/3*220
ti=15/8*220
do1=2*220
re1=2*9/8*220
mi1=2*5/4*220
fa1=2*4/3*220
so1=2*3/2*220
la1=2*5/3*220
```

(continues on next page)

(continued from previous page)

```
ti1=2*15/8*220
do2=2*2*220
scale=[do,re,mi,fa,so,la,ti,do1]
```

```
def play(song):
    song=np.array(song)
    framerate = 44100
    t = np.linspace(0, len(song) / 2, round(framerate * len(song) / 2))[:-1]
    song_idx = np.floor(t * 2).astype(int)
    data = np.sin(2 * np.pi * song[song_idx] * t)
    return Audio(data, rate=framerate, autoplay=True)
play(scale)
```

```
<IPython.lib.display.Audio object>
```

5) Can you name that tune?

Note

Please visit the web version of this section to engage with this interactive resource.

```
tune= [so, so , la, la, so, fa,mi,rest,so, so , la, la, so, fa,mi,rest,so,so,la,ti,
 ↪do1,do1,re1,re1,ti,la,ti,la,so,so,la,ti,do1,do1,ti,la,so,so,rest,rest,la,la,so,fa,
 ↪mi,mi,rest,rest,so,so,do,fa,mi,mi,re,re,do,do,do,do,rest,rest]
play(tune)
```

```
<IPython.lib.display.Audio object>
```

```
from IPython.display import YouTubeVideo
YouTubeVideo('L4PA-MFSM34')
```

"We Shall Overcome (Live Recording)" facebook video, uploaded by SOS from the Kids, https://www.facebook.com/watch/?v=234581491372019

Exercise

Exercise

Demonstrate how to create a new tune.

2.8 JNB Lab Solutions

```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
```

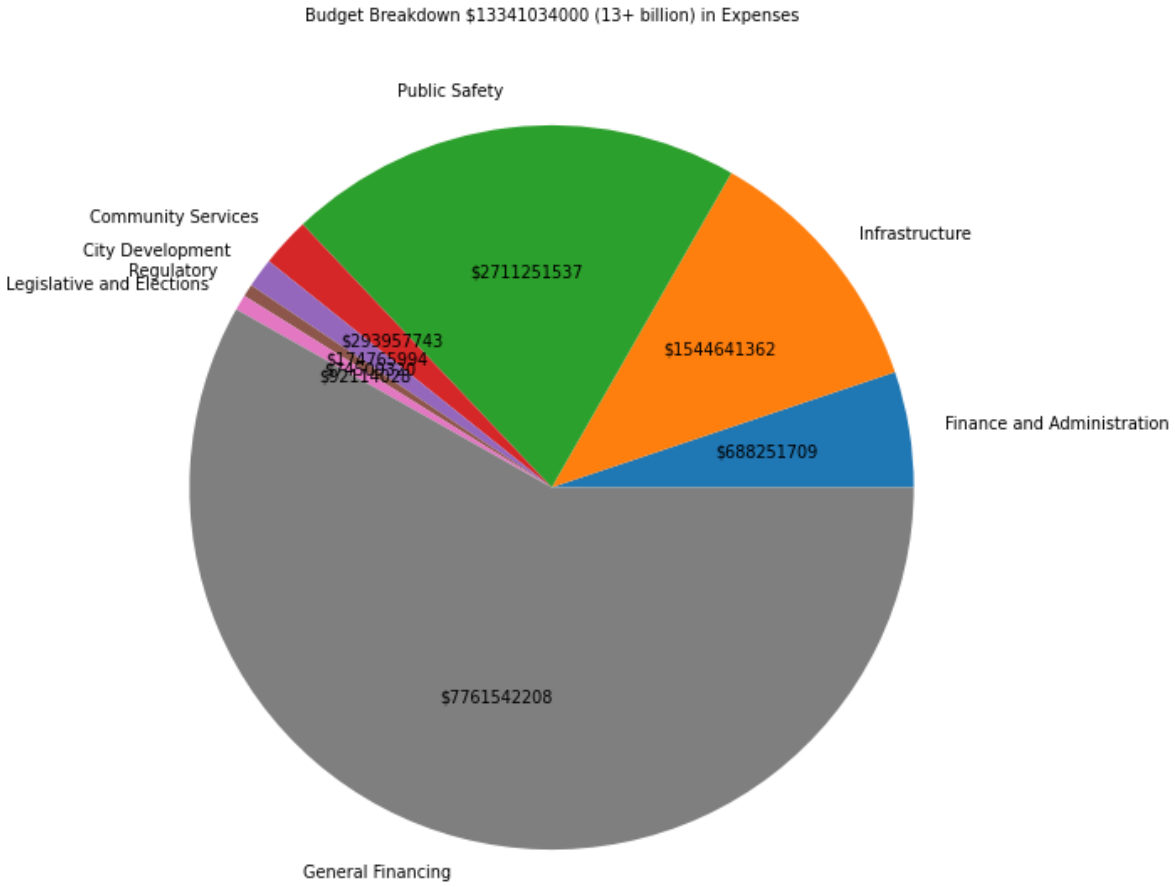
Demo 1 Exercise

```
budget=pd.read_excel('ChicagoBudget.xlsx')

fig, ax = plt.subplots(figsize=(7,7)) #you can adjust the figsize  (5,5)=(length,
 ↪width)
```

(continues on next page)

(continued from previous page)

```
plt.rcParams['font.size'] = 7  #fontsize
budget_items = budget["EXPENSE"] #categories
budget_amounts = budget["2023 BUDGET"] #amounts
total=sum(budget_amounts)
ax=plt.pie(budget_amounts,labels=budget_items,autopct=lambda p: '${:.0f}'.format(p *␣
 ↪total / 100)) #make pie chart  autopct='%1.0f%%'
plt.gca().set_title('Budget Breakdown $'+str(total)+' (13+ billion) in Expenses',
 ↪size=7) #add a title
fig.savefig('Budget.png') #save the piechart to a file Budget.png
```

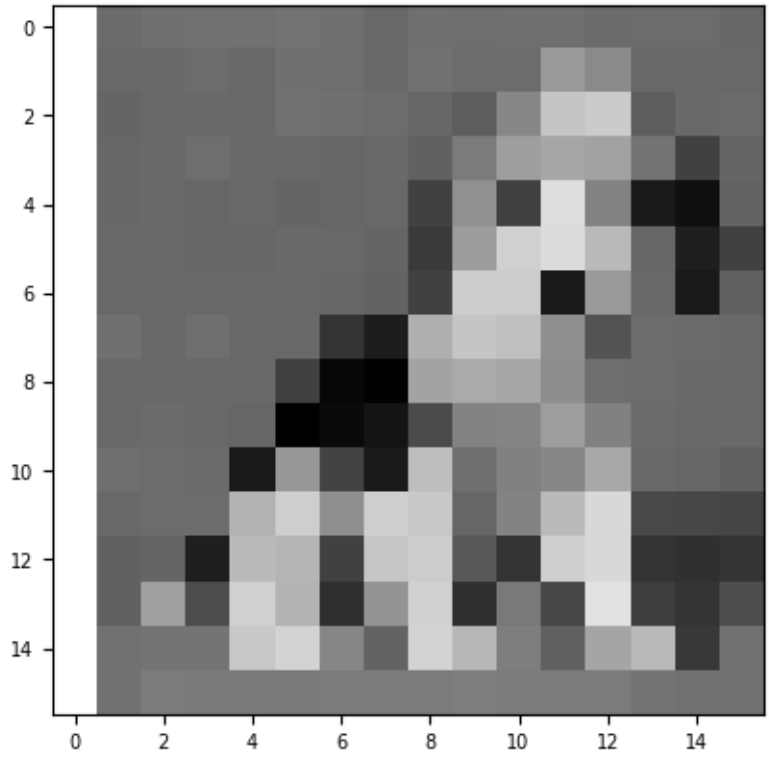


Demo 2 Exercise

```
# PACKAGE: DO NOT EDIT THIS CELL
%matplotlib inline
from ipywidgets import interact
import cv2, os
```

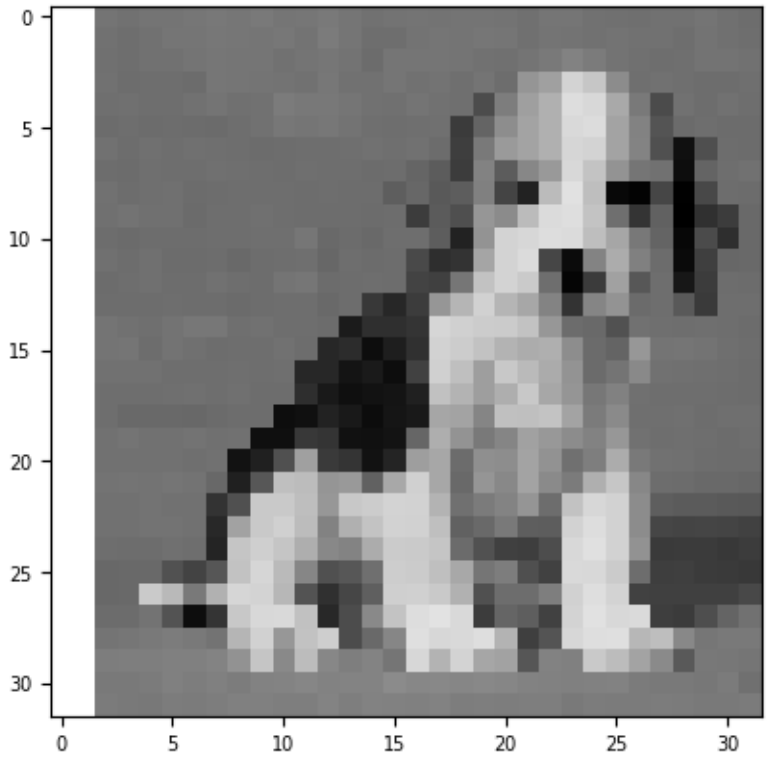
```
makepixelimage("images", 16) #16x16 image
```

```
<matplotlib.image.AxesImage at 0x1874f628a90>
```

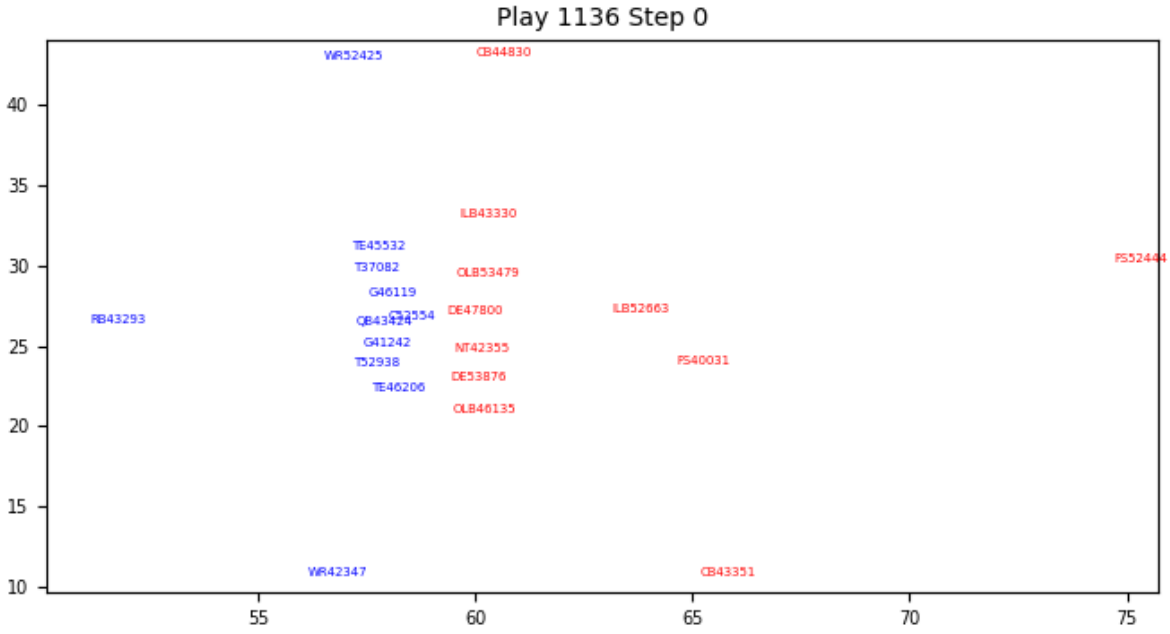



```
makepixelimage("images", 32) #32x32 image
```

```
<matplotlib.image.AxesImage at 0x1874f7f8550>
```



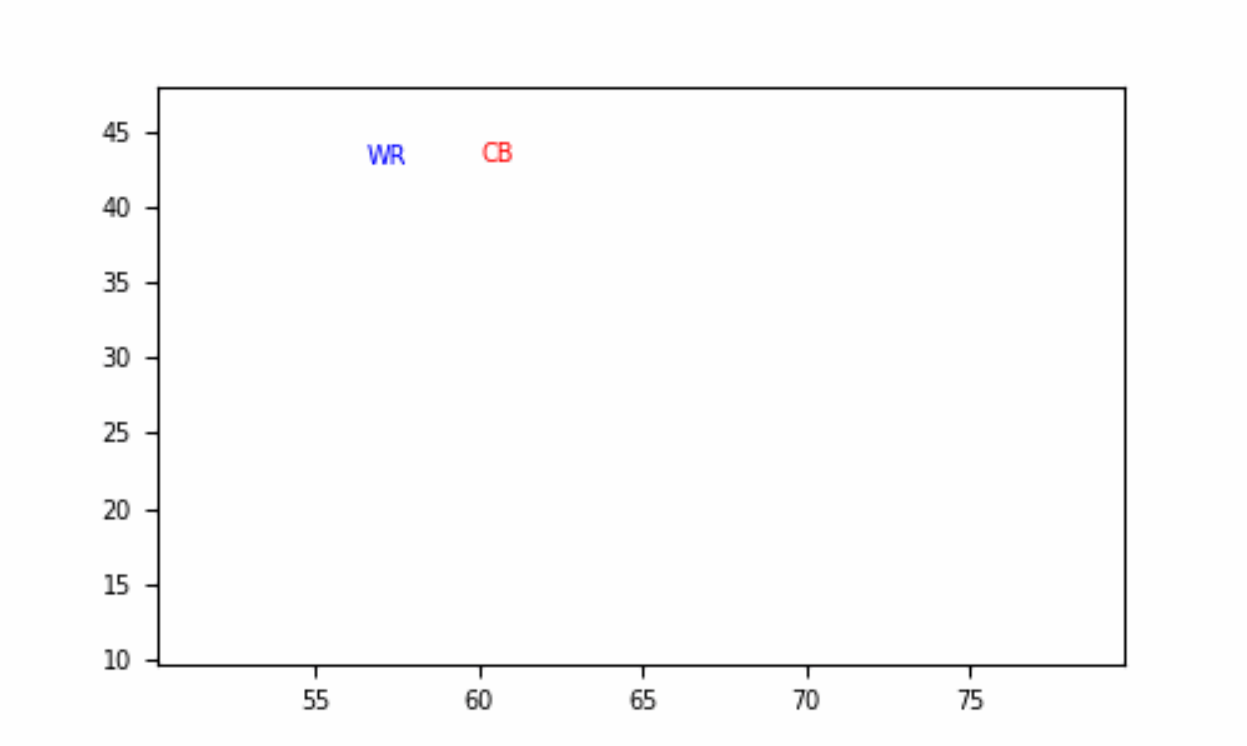
DEMO 3 Exercise



The wide receiver at the top is 52425 and cornerback is 44830.

Note

Please visit the web version of this section to engage with this interactive resource.



Demo 4 Exercise

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

Demo 5 Exercise

Note

Please visit the web version of this section to engage with this interactive resource.

```
jingle= [mi, rest,mi ,rest, mi, rest,rest, mi,rest, mi,rest,mi,rest,rest,mi, mi,so ,
 ↪so, do,do, re,re, mi,mi]
play(jingle)
```

```
<IPython.lib.display.Audio object>
```

CHAPTER 3

PYTHON PROGRAMMING GUIDE - Thomas VanDrunen, Yiheng Liang

Note

Computer scientists Dr. VanDrunen (Wheaton College) and Dr. Liang (University of Georgia) introduce the Python programming language and how it is used to process data.

3.1 Introduction

This chapter is designed to help us transition from the use of JNBs in pre-college outreach to the use of JNBs within college curriculum. Section 2 give a guide to Python programming to help navigate the reader through the remainder of the book. Section 3 covers the basics of working with data, including the use of the pandas and matplotlib libraries.

3.2 Introduction to Python

This chapter introduces programming concepts and how those concepts are expressed in the Python programming language. Keep in mind that *learning to program* and *learning the Python programming language* are different things. Learning to program involves learning to break problems down into computational steps and synthesizing those steps into an algorithm, independently of what programming language, if any, one is using. To learn a programming language is to learn how those computational steps are expressed in the syntax and idioms of a specific coding system.

Learning to program and learning a programming language both take time and effort. A chapter like this isn't enough to teach you everything about Python if you already know how to program in another language, and it certainly isn't enough to teach you how to program if you haven't programmed before. Many good books and online tutorials exist for those purposes. What we hope to accomplish in this chapter is highlight basic facts about programming and the Python programming language so that you know what to anticipate in the chapters that follow. This way you can begin to follow and make changes to the code portions of the notebooks. Obsvering the code in this book can be your first step towards writing programs yourself.

3.2.1 Expressions, types, variables, and statements

The fundamental building block of code is an *expression*, which is a construct in the programming language that evaluates to a value. We can put an expression in a code cell in a notebook like this, and the Python interpreter will read that expression, evaluate it, and display the result. There are many kinds of expressions, but the easiest to understand are mathematical expressions, since they look mainly the same as they do in mathematical writing. Here are a few examples.

```
5
```

```
7+3
```

```
(27-9)/17
```

Values have different *types*. A type is a set of values associated together by how they are stored in computer memory and what operations are defined for them. Integers are real numbers and are represented by different types in Python. We can see the type of an expression by using the *type* function.

```
type(3+7)
```

```
type(24/7)
```

The type `int` is obviously short for *integer*. The real number type is called `float`–that is, *floating point*–in reference to how fractional values are stored in computer memory. Notice that in these two cases, the type itself is the value: the result of evaluating `type(3+7)` is the *type* of the result of `3+7`, not the result of `3+7` itself.

Not all values are numeric. Other type of values include `str` (*string*, for text values) and `bool` (*boolean*, for logical values).

```
'jupyter notebook'
```

```
type('jupyter notebook')
```

```
3 < 7
```

```
type(3 < 7)
```

Values can be stored in *variables* so that they can be reused.

```
x = 7 + 3
x * 12
```

In the first line of that cell, `7 + 3` is an expression, as in earlier examples, but the line as a whole, `x = 7 + 3`, is not an expression but a *statement*. This means it does not result in a value but rather has a *side effect*, which means it makes the Python interpreter do something like display a plot, print something to the output, or, in this case, store a value in a variable. This is important because it means `x = 7 + 3` is not a *comparison* (Is *x* equal to `7 + 3`?) but an *assignment* (Store the result of `7+3` in variable *x*!). To test if two expressions evaluate to the same value, we use == instead of =.

```
x == 12
```

For a lot of purposes your knowledge of variables from math can guide you in understanding variables in Python or other code. But the difference between comparison and assignment bring up an important difference: just as variables can be assigned, they can also be *reassigned*. In fact, they can be used and reassigned in the same statement.

```
x = x + 1
x
```

Mathematically $x = x + 1$ makes little sense, apart from always being false. (Can you think of any algebraic structure in which an element is equal to itself plus the multiplicative identity? I can't.) But in programming it means, "Take the value that *x* used to be, add 1 to that value, and store that value back in variable *x*. To put a fancy word to it, variables in Python are *mutable*.

As a final observation about expressions and statements, notice that in each of the cells so far the output has been the value that results from the last expression. If we want to display more information, we can use the `print()` function, which has the side effect of displaying something to output. This function takes a string as its input. We can make a bigger string from smaller strings using +, which in the context of strings does *string concatenation*. Most non-string values can be turned to strings using the `str()` function.

```
print('At first x is ' + str(x))
x = x + 1
print('Now x is ' + str(x))
x = x * 25
print('Finally, x is ' + str(x))
```

3.2.2 Conditionals and while loops

If a computational task is complicated enough that we want to use a computer for it, then it almost certainly requires some sort of decision-making and repetition. The most basic construct for making a decision in an algorithm is a *conditional*, which uses the Python keyword `if`.

```
if x < 12 :
    print('Hello')
```

Since `x` (from earlier) is *not* less than 12, the print statement wasn't executed, and this cell had no output. We can make this into a decision between alternatives using `else`.

```
if x < 12 :
    print('Hello')
else :
    print('Aloha')
```

We can make a multiway branch by using any number of `elif` (short for `else if`) clauses in our conditional statement.

```
if x < 12 :
    print('Hello')
elif x < 100 :
    print('Howdy')
elif x < 350 :
    print('Ni hao')
else :
    print('Aloha')
```

Repetition is implemented using a loop, which executes a statement (the *body* of the loop) as long as a condition (the *guard* of the loop) is true. Each execution of the body is an *iteration* of the loop.

```
y = 10
while y > 0 :
    print('There are ' + str(y) + ' iterations left.')
    y -= 1
    print('(Deep breath)')
```

Notice several things in this example: The body of the loop consists of three statements; however, these three statements are treated as one *compound statement* because they are all indented together. The statement `y -= 1` is a convenient

shorthand for `y = y - 1`. We can similarly use `y += 1` to mean increment *y* by one, etc. Finally, the loop did not stop as soon as *y* was no longer greater than 0 (when `y -= 1` was executed the final time) but rather when it finished the iteration, printing `(Deep breath)` one last time. This is because the guard is checked only between iterations—it is not continuously checked while the body is executed.

If we wanted to quit the loop in the middle of an iteration, we can do that using a `break` statement, which we can put inside a conditional. In the following example, we use this instead of the guard and simply make the guard itself to be `True`.

```
y = 10
while True :
    print('There are ' + str(y) + ' iterations left.')
    y -= 1
    if y <= 0 :
        break
    print('(Deep breath)')
```

To see how well you follow this, make sure you can explain, *(1)* Why did we invert the guard from `y > 10` to the `y <= 10`? and, *(2)* Why didn't the output end with `(Deep breath)` as in the original loop?

3.2.3 Functions

You already know about functions from mathematics. With just a little bit of Python syntax, you can make mathematical functions that fit right into the Python coding we've seen. Let's take

$f(x, y) = 2x^2 - 3x + \frac{y}{x}$

If we replace the = for a `:` and throw in the Python keywords `def` and `return` (because we are *defining* a function to *return* a certain value as its result), we get

```
def f(x, y) :
    return 2 * x**2 - 3 * x + y/x
```

Now we can *call* or *apply* the function to some *parameters* (or arguments), which all together makes an expression. (The `**` operator does exponentiation.)

```
f(5, 3)
```

That's well and good mathematically. But in programming, a function can be much more than that: it's a way to package up a chunk of code, give the chunk of code a name, specify a variable to hold the input to that chunk of code, and indicate what that chunk of code's output is, based on what is computed from the input when the chunk of code is executed. This allows that chunk of code to be executed many times from any context. (And we promise that we'll never use the phrase *chunk of code* from this point on.)

```
def greatest_common_divisor(a, b) :
    while b > 0 :
        r = a % b    # The % operator is called "modulo" or "mod". It compute a␣
↪remainder
        a = b
        b = r
    return a
```

```
greatest_common_divisor(24, 18)
```

```
greatest_common_divisor(100, 75)
```

```
def mad_lib(exclamation, adverb, plural_noun, verb):
    print('"' + exclamation + '!" he said ' + adverb + '. "I didn\'t know ' + plural_
 ↪noun +
          ' could ' + verb + '."')
```

```
mad_lib('Wow', 'warily', 'cardinals', 'yodel')
```

3.2.4 Anonymous (lambda) Functions

In Python, functions are *values*, just like integers, floats, bools, and strings. This means that we can store them in variables. The following statement assigns the variable `g` to store the same function as is already referred to by function `f`:

```
g = f
```

The result is that `g` is an alias for `f`:

```
g(12, 7)
```

Since functions are values, we can make a function value without a name at all—an *anonymous* function. The syntax for an anonymous function begins with the keyword `lambda` which, for historical reasons, is used to introduce function definitions; followed by a list of parameters; a colon; and then an expression for computing the value to be returned. Here is a simple example that encapsulates the expression x^2 as a function, but does not give that function a name.

```
lambda x:x**2
```

Since this value is a function, it can be called. Here the anonymous function is again, but this time immediately applied to a parameter:

```
(lambda x:x**2)(-3)
```

Make sure you understand that `lambda` is not the name of the function—it has no name at all. Moreover, there is no keyword `return`—the entire body of the function is simply the expression whose value is to be returned.

We can store the result of a lambda expression in a variable. This gives the function a name, so once we have done that the function is no longer anonymous, and we can call the function by name as we usually would.

```
h = lambda x:x**2
h(-3)
```

Some find the usage above to be a compact way to introduce a new, simple function, but it doesn't have any real advantage over the following (you can judge for yourself what's more readable or intuitive):

```
def h(x) : return x**2
```

The real advantage to *functions as values* is that functions can be passed to other functions and, if useful, returned from other functions. Take the idea of function composition. If you have two functions f and g, you can compose them into a new function $g \circ f$ that feeds the result of f into function g as its parameter. (In order for this to work, the codomain/return type of f must match the domain/parameter type of g.)

$$g \circ f(x) = g(f(x))$$

For example, if $f(x) = x^2$ and $g(x) = 3x$, then $g \circ f(x) = 3x^2$.

The following function is the equivalent of the function composition operator, $\circ$, in that it takes two functions are returns a new function made by composing the two given functions.

```
def compose(g, f) :
    return lambda x : g(f(x))
def a(x) : return x**2
def b(x) : return 3 * x
b_a = compose(b, a)
b_a(5)
```

In writing `compose`, we used both the regular function form with `def` and this new form using `lambda`. This is because even though `compose` makes and returns an anonymous function, it itself is not anonymous.

3.2.5 Lists and other indexables and iterables

For collections of data, we need more than variables that hold a single value. The code in the chapters that follow makes use of various specialized *data structures*, but for right now we'll observe how to use the simplest *container* in Python, which is the *list* data structure. A list is an ordered sequence of values. We can express a list in Python by, well, *listing* a bunch of values, separated by commas and surrounded by square brackets. More precisely we don't need to list values but rather expressions that the values result from.

```
[5, 7 + 8, "geoduck", 14 < 3, 12 / 5]
```

A list also is a value, just of a different type. We can store list values in variables.

```
z = [5, 7 + 8, "geoduck", 14 < 3, 12 / 5]
type(z)
```

Since a list is a linear sequence, we can refer to and retrieve a list's components by *index*. Python lists are indexed from 0, which means that what normal people might call the "first" element is, for our purpose, the "zeroth" element. The `len()` function tells us the number of elements in the list.

```
print(z[0])
print(z[1])
print(z[len(z) - 1])
```

Notice that zero-based indexing means that the last element is in position one-less-than-the-length. The length of the list itself is not a valid index.

```
z[len(z)]
```

The most straightforward interpretation of a list for mathematical purposes is as a sequence of numbers. Using Python pieces we know so far, we can write a function that computes simple statistics for a list:

```
def simple_stats(xx) :
    min = xx[0]  # The minimum value we've seen so far
    max = xx[0]  # The maximum value we've seen so far
    sum = xx[0]  # The sum of all the numbers we've seen so far
    i = 1        # What does the variable i mean?
    while i < len(xx) :
        # x[i] is the current number we're looking at.
        # Add it to our running sum
        sum += xx[i]
        # If it's less than the minimum-so-far, then it's the new minimum-so-far
        if xx[i] < min :
            min = xx[i]
        # Similarly with maximum
        if xx[i] > max:
```

(continues on next page)

(continued from previous page)

```
            max = xx[i]
        i += 1
    print('The sum of the numbers is ' + str(sum))
    print('The average of the numbers is ' + str(sum/len(xx)))
    print('The greatest number is ' + str(max))
    print('The least number is ' + str(min))
```

```
simple_stats([2,6,9,12,17,1,25,14])
```

The function `simple_stats()` doesn't have a return statement. This means when it is called, the call doesn't evaluate to a value—it just has side effects, namely printing results to output.

To answer our question, "What does the variable `i` mean?", it means several things:

- `i` is the number of values we've processed so far.
- `i` is the index of the next value we'll look at.
- `min` is the minimum of all the values in indices 0 (inclusive) through `i` (exclusive).
- `i` is one more than the number of iterations completed.

We can also modify a list by writing to a position or add to it by using the `append()` function.

```
z[3] = 'Yoda'
z
```

```
z.append(42)
z
```

You'll notice that `append()` is used differently from other functions we've seen since the list we're appending appears to the left of the function name, separated by a dot, rather than in the parentheses. The value being appended is in the parentheses. It's a different kind of function, but that's a story we'll have to come back to later.

But the real power of Python indexing is that you can specify a range (a sublist or a substring) or count from the back of the list using negative indices. See how well you can infer the rules of indexing from these examples.

```
aa = ['A','B','C','D','E','F','G']
```

```
aa[-1]
```

```
aa[-3]
```

```
aa[2:5]
```

```
aa[1:5]
```

```
aa[:3]
```

```
aa[3:]
```

```
aa[-3:]
```

```
aa[:-3]
```

The fact that lists have components that can be referred to using square brackets means that lists are *indexable*. Strings are also indexable.

```
ss = 'Jupyter Notebook'
ss[5]
```

Lists and strings (and many other Python containers) share another useful property: They are *iterable*, which means they can be used in special constructs designed to *iterate* over the components. One construct is a *for loop*, an alternative to the *while loop* that works on iterables:

```
for a in z:
    print(a)
```

The way to interpret this is that the variable `a` *ranges over the values* in the list `z`. In each iteration, `a` takes on a different value in the list. Observe the usefulness of iterables in this simpler version of our `simple_stats()` function from before, this time just computing the average.

```
def average(xx) :
    sum = 0
    for x in xx :
        sum += x
    return sum / len(xx)
```

```
average([4,72,3,10,5,63])
```

To see this on strings (the function `upper()` converts text to uppercase):

```
def cheer(name) :
    for c in name :
        c = c.upper()
        if c in 'AEFHILMNORSX' :  # Letters whose names begin with a vowel sound
            print('Give me an ' + c)
        else :
            print('Give me a ' + c)
    print("What's that spell?")
    print(name)
```

```
cheer('Python')
```

We snuck in some new Python in that last example—using `in` to see if the character `c` occurs in our list (actually a string) of letters whose names begin with a vowel sound. It turns out that `in` can be used both in a for loop to iterate over an iterable, but also in an expression to test whether an element exists somewhere in an iterable.

One more iterable is particularly useful: the `range()` function, which returns an iterable over numbers in the range indicated in the parameters to the function. If you give the function one number, it is interpreted as an exclusive upperbound, with 0 as the inclusive lower bound:

```
for i in range(4) :
    print('Say hello, ' + str(i))
```

Or you can specify both an (inclusive) lower bound and an (exclusive) upper bound.

```
print('My favorite primes are')
for i in range(11,14) :
    print(i)
print('... well... except for 12.')
```

Finally, you can specify an increment:

```
for i in range(2, 9, 2) :
    print(i)
print('Whom do we appreciate?')
for i in range(5, 0, -1) :
    print('T-minus ' + str(i) + ' seconds')
print('Lift off')
```

Together with `len()`, ranges provide us with another idiom for iterating over the values in a list, if for some reason we don't want to use the list as an iterable directly. For example, suppose we have a list of the coefficients of the polynomial $3 + 4x - 7x^2 + 5x^3$. The following function evaluates a polynomial, given a list of coefficients and an x value:

```
def evaluate_polynomial(coefficients, x) :
    result = 0
    for i in range(len(coefficients)) :
        result += coefficients[i] * x**i
    return result
```

```
evaluate_polynomial([3,4,-7,5], 2)
```

How many meanings can you find for the variable `i`?

3.2.6 Thinking through algorithms

Here is a function that takes a list `yy` and another value `x` and determines the first index in `yy` that contains `x`. If `x` doesn't exist anywhere in the list, then the function returns the special value `None`.

```
def bounded_linear_search(yy, x) :
    i = 0
    found_it = False
    # As long as we have more positions to inspect and we haven't found what we're␣
 ↪looking for
    while i < len(yy) and not found_it :
        # If i's the one we're looking for...
        if yy[i] == x :
            # ... great! We found it.
            found_it = True
        # Otherwise ...
        else :
            # move on to the next one, if any.
            i += 1
    if found_it :
        return i
    else :
        return None
```

```
zz = [4,2,8,7,12,16,1,9,11]
print(bounded_linear_search(zz, 7))
print(bounded_linear_search(zz,5))
```

Here is another function that does something similar, but *assumes the list is sorted from least to greatest*. Because of this assumption it doesn't start at the beginning but instead looks in the middle. Depending on whether the value in the middle is too big or too small, it "throws away" either the top half or the bottom half of the list (or, if the value is just right, it returns that index). To do this, the algorithm keeps track of an upper and lower bound of the range of indices in the list

where the searched value could be—in other words, it has determined that the searched value *can't* be *outside* that range. (The `//` operator does *integer division*. For positive integers, this means rounding down to the nearest integer. We do this because indices need to be integers.)

```
def binary_search(yy, x):
    lower_bound = 0
    upper_bound = len(yy)
    mid = (lower_bound + upper_bound) // 2
    # As long as the range is more than just one position...
    while upper_bound - lower_bound > 1 :
        # ...if the midpoint is the value we're looking for...
        if yy[mid] == x :
            # ...then collapse the range to include just that position
            lower_bound = mid
            upper_bound = mid+1
        # ...otherwise, if the midpoint is too big...
        elif yy[mid] > x :
            # ...then throw away the top half of the range
            upper_bound = mid
        # ...otherwise, the only option left is that the midpoint is too small, so..
        else :
            assert yy[mid] < x
            # ...throw away the bottom half of the range.
            lower_bound = mid + 1
        # Recompute the midpoint index
        mid = (lower_bound + upper_bound) // 2

    # Check that the range isn't empty and that the midpoint is what we're looking for
    if upper_bound - lower_bound == 1 and yy[mid] == x :
        return mid
    else :
        return None
```

(`assert` means "I am certain of this proposition, may my program crash if I am wrong.")

```
zz = [1,4,5,17,19,25,33,34,37,40,44,58,63]
print(binary_search(zz, 17))
print(binary_search(zz, 43))
```

How do these algorithms differ? Is one better than the other? We have already observed that the second algorithm makes a stronger assumption about its input than the first one does. It takes a very different strategy, which wouldn't be correct if we didn't make that stronger assumption. Thinking carefully about code comes down to two things: Correctness and efficiency.

We can reason about an algorithm's efficiency by identifying what must be true of the state of the computation at each step. The *state of the computation* is captured by the value of the variables, and this ties in with what we've already alluded to when talking about the meaning of variables. What does the variable `i` mean in `bounded_linear_search`? Similar to `simple_stats`,

- `i` is the number of values we've processed so far.
- `i` is the index of the next value we'll look at.
- `i` is the number of iterations completed.

But especially `i` represents the algorithm's progress in searching for the indicated value `x` (call it the *search value*). That is, `i` marks the boundary between two sections of the list: The indices less than `i` are *known not to contain* the search value; if the search value is anywhere in the list at all, it must be in an index `i` or greater.

- For all indices j, where $0 \leq j <$ `i`, `yy`$[j] \neq$ `x`

(We use typewriter font for variables that actually occur in the code. Since j is not a variable in the code, it gets the usual italic font.)

Here's a new version of `bounded_linear_search` with the comments taken away but where we assert this last claim about `i`. Practically speaking this defeats the whole purpose of `bounded_linear_search` since this check does the same work as the entire algorithm, but we're doing it here for demonstration purposes.

```
def bounded_linear_search(yy, x) :
    i = 0
    found_it = False
    while i < len(yy) and not found_it :
        assert not x in yy[:i]    # x is nowhere in y between 0 and i
        if yy[i] == x :
            found_it = True
        else :
            i += 1
    if found_it :
        return i
    else :
        return None
```

```
zz = [4,2,8,7,12,16,1,9,11]
print(bounded_linear_search(zz, 7))
print(bounded_linear_search(zz,5))
```

For `binary_search`, the variables `lower_bound` and `upper_bound` define the region in the list where the search value might be. Here's a new version that asserts that the search value isn't anywhere outside that range.

```
def binary_search(yy, x):
    lower_bound = 0
    upper_bound = len(yy)
    mid = (lower_bound + upper_bound) // 2
    while upper_bound - lower_bound > 1 :
        assert not x in yy[:lower_bound] and not x in yy[upper_bound:]
        if yy[mid] == x :
            lower_bound = mid
            upper_bound = mid+1
        elif yy[mid] > x :
            upper_bound = mid
        else :
            assert yy[mid] < x
            lower_bound = mid + 1
        mid = (lower_bound + upper_bound) // 2

    if upper_bound - lower_bound == 1 and yy[mid] == x :
        return mid
    else :
        return None
```

```
zz = [1,4,5,17,19,25,33,34,37,40,44,58,63]
print(binary_search(zz, 17))
print(binary_search(zz, 43))
```

But the real difference between these two algorithms can be seen by looking at their efficiency. The simplest way to observe the computational cost of an algorithm is to count the number iterations of its loop (assuming it's only got one…). In this third version of `bounded_linear_search`, we display the number of iterations before the function returns.

```
def bounded_linear_search(yy, x) :
    i = 0
    found_it = False
    while i < len(yy) and not found_it :
        assert not x in yy[:i]    # x is nowhere in y between 0 and i
        if yy[i] == x :
            found_it = True
        else :
            i += 1
    print(str(i) + ' iterations')
    if found_it :
        return i
    else :
        return None
```

Of course, the number of iterations depends on when the algorithm finds the search value, and the *worst case* in terms of cost is if it doesn't occur in the list at all. Let's observe the number of iterations if we search for a value in increasingly large lists which don't contain that value.

```
bounded_linear_search([5]*8, 9)
bounded_linear_search([5]*16, 9)
bounded_linear_search([5]*32, 9)
bounded_linear_search([5]*64, 9)
bounded_linear_search([5]*128, 9)
bounded_linear_search([5]*256, 9)
```

The number of iterations equals the size of the list, but perhaps you could have predicted that, since each iterations inspects one position in the list. Let's do a similar experiment on `binary_search`. Since that function doesn't have a variable like `i` that tracks with the number of iterations, we need to add such a variable.

```
def binary_search(yy, x):
    lower_bound = 0
    upper_bound = len(yy)
    mid = (lower_bound + upper_bound) // 2
    i = 0
    while upper_bound - lower_bound > 1 :
        if yy[mid] == x :
            lower_bound = mid
            upper_bound = mid+1
        elif yy[mid] > x :
            upper_bound = mid
        else :
            assert yy[mid] < x
            lower_bound = mid + 1
        mid = (lower_bound + upper_bound) // 2
        i += 1

    print(str(i) + ' iterations')
    if upper_bound - lower_bound == 1 and yy[mid] == x :
        return mid
    else :
        return None
```

```
binary_search([5]*8, 9)
binary_search([5]*16, 9)
binary_search([5]*32, 9)
binary_search([5]*64, 9)
```

(continues on next page)

(continued from previous page)

```
binary_search([5]*128, 9)
binary_search([5]*256, 9)
```

Not only is this algorithm starkly more efficient, we can mathematically quantify the difference in efficiency. The worst-case number of iterations of `binary_search` tracks with the base-two logarithm of the length of the list. And this, too, makes sense: In each iteration of `binary_search` we throw away half of the remaining range of the list, and so it takes $\log_2 n$ steps to whittle a list of length n down to a trivial range.

(We said the number iterations "tracks with" the base-two logarithm of the length of the list. How does the number of iterations relate to the length of the list precisely?)

3.2.7 Python libraries

In practical programming it rarely makes sense to develop all your code from scratch. Many common problems have already been solved, and their solutions have been collected for reuse. *Libraries*, organized collections of reusable code, act as extensions of the programming language itself, and an important part of mastering any programming language is learning the libraries that come standard with the language and the widely-used libraries that have been developed by the language's community of users. This section explains the fundamentals of libraries and related concepts like modules and packages by showing how to manage code with your own libraries and demonstrating parts of Python's standard library. Later sections highlight other libraries that are useful to know for other chapters in this book.

Suppose that you frequently find yourself writing code that makes use of geometric formulas, such as the circumference or area of a circle or the volume of a sphere or cylinder. As you know by now, the natural thing to do is not to replicate the code for each formula every time you need it but to encapsulate the formula into a function—not only for convenience (avoid re-coding the formula) but also for readability (the code now has a name).

But if you have several notebooks or applications that make use of the same formulas, then pasting those function definitions into cells in each new notebook creates unnecessary clutter. Instead, collect those function definitions into a separate Python file.

The folder that contains this notebook also has a file `geometry.py` consisting of the definitions for `circumference_circle`, `area_circle`, `volume_sphere`, and `volume_cylinder`. Note that this is not a *notebook*, which would have a `.ipynb` file extension. You should be able to look at the file in your browser but not execute its code that way.

We can, however, make the code there available for use in this notebook. First, we *import* the code in that file:

```
import geometry
```

Now we can call one of the functions.

```
geometry.volume_sphere(3.8)
```

Notice that we call the function by the *qualified name* `geometry.volume_sphere` instead of the simple name `volume_sphere` as it appears in the file itself. The prefix `geometry.` indicates that `volume_sphere` is a part of the module `geometry`. Formally, a *module* is a collection of code contained in a Python file.

There are a variety of ways to use `import` statements. For example, the following import statement grabs only the function `area_circle`, and the subsequent code doesn't need to qualify the name when it is used.

```
from geometry import area_circle
```

```
area_circle(5.7)
```

If you have a large number of related modules, it's natural that you would collect them into one or more folders in the file system. In Python terminology, a folder of Python files constitutes a *package*, which, in turn, can have subpackages in subfolders. A large collection of modules, often organized into many packages, is called a *library*.

The *standard library* comprises the packages that are an integral part of any Python installation. You can see what the standard library offers by perusing its documentation or working through any comprehensive Python tutorial. For our purposes, consider these two examples.

The `math` package provides an abundance of functions and constants for familiar mathematical tasks.

```
import math
```

```
# least common multiple of a collection of integers
print(math.gcd(120, 96))
# 'floor' of (greatest integer less than) a float
print(math.floor(26.7))
# arctangent (result in radians)
print(math.atan(2.3))
# e (Euler's number) raised to a given power
print(math.exp(5.8))
# tau (mathematical constant equal to two pi)
print(math.tau)
```

The `random` package contains functions for generating random numbers.

```
import random
```

```
# Pick an integer uniformly distributed within a given range.
print(random.randint(3,17))
# Pick a float from a Gaussian distribution with a given mean and standard deviation
print(random.gauss(.6, 3.5))
xx = [2,7,3,8,14,9,63]
# Permute a sequence
random.shuffle(xx)
print(xx)
# Pick an element from a sequence
print(random.choice(xx))
```

The packages in Python's standard library are extensive, and yet are just the beginning of the tools available in libraries published by other organizations. In contrast to the basic mathematical functions provided by the `math` package, the `scipy` library from scipy.org supports more advanced operations like fast Fourier transform, solving systems of linear equations, and interpolation.

3.2.8 Classes and objects

Programming is all about *data* and *operations on data*. We have seen that types define categories of data and that functions encapsulate algorithms that operate on data. Types range from primitive types like `int` and `float` to complicated data structures likes `lists`.

In Python (and most other modern programming languages), *classes* are a language feature by which a programmer can define new types. Moreover, classes allow data and functionality to be encapsulated together.

Suppose we want to make a type for complex numbers (actually Python has a complex number type, but suppose we're hard-core do-it-yourselfers). A value of this type is composed of two parts, the so-called "real" and "imaginary" parts, each of which are float-valued. The following `Complex` class defines a type for values like this.

```
# Class to define a complex number type, first try
class Complex :
    def __init__(self, a, b) :
        self.real_part = a
        self.imaginary_part = b
```

Before explaining this code, let's see how it is used to make the value $3.5 + 2.7i$.

```
z = Complex(3.5, 2.7)
print(z)
print(type(z))
print(str(z.real_part) + ' + ' + str(z.imaginary_part) + 'j')
```

Here is how to understand all of this:

- The variable `z` stores a value whose type is `Complex`.
- That value has two components called `real_part` and `imaginary_part`, and those components each have type `float`.
- We use `.` to retrieve a component—hence `z.real_part` means "retrieve the component called `real_part` from the `Complex` value stored in variable `z`."
- The class `Complex` has a special function `__init__` that initializes the components of a `Complex` value. Inside the `__init__` function, the value whose components are being initialized is referred to by the parameter `self`.
- A value of a class type is called an *object*. It is also an *instance* of that class. Making a new instance of a class an initializing its components is called *instantiating* that class.
- The expression `Complex(3.5, 2.7)` was used to instantiate the `Complex` class. This not only made a new instance of the class but also called the `__init__` function. The new instance was passed to `__init__` as `self` and the values `3.5` and `2.7` were passed as the other parameters to `__init__`.

Now that we've made this class and an instance of the class, we can make functions for complex operations

```
def magnitude(z) :
    return (z.real_part**2 + z.imaginary_part**2)**.5

def add_complex(z, w) :
    return Complex(z.real_part + w.real_part, z.imaginary_part + w.imaginary_part)

def mult_complex(z, w) :
    return Complex(z.real_part * w.real_part - z.imaginary_part * w.imaginary_part,
                   z.real_part * w.imaginary_part + z.imaginary_part * w.real_part)
```

That works just fine for a simple class like `Complex`. But what makes objects distinctive is that they can contain not only variables but also functions. A function that is a component of a class is called a *method*. In this second version of the `Complex` class we include methods like the functions defined in the previous cell.

```
# Class to define a complex number type, first try
class Complex :
    def __init__(self, a, b) :
        self.real_part = a
        self.imaginary_part = b

    def magnitude(self) :
        return (self.real_part**2 + self.imaginary_part**2)**.5

    def add(self, other) :
```

(continues on next page)

(continued from previous page)

```
        return Complex(self.real_part + other.real_part, self.imaginary_part + other.
↪imaginary_part)

    def mult(self, other) :
        return Complex(self.real_part * other.real_part - self.imaginary_part * other.
↪imaginary_part,
                       self.real_part * other.imaginary_part + self.imaginary_part *␣
↪other.real_part)
```

Since the methods are components—or *attributes*—of the *Complex* objects, we refer to them using `.` notation:

```
z = Complex(2.1, 4.7)
w = Complex(-5.2, 8.5)
y = z.add(w)
print(str(y.real_part) + ' + ' + str(y.imaginary_part) + 'i')
```

In the expression `z.add(w)`, notice that `add` has only one actual parameter, `w`, which in fact is assigned to the formal parameter `other`. The parameter `self` is assigned the object that the method is called on, `z`.

The variable components of a class (called *data attributes* or *instance variables*) can be modified. This mutates the state of the object.

```
class Hummingbird :
    def __init__(self) :
        self.alive = True
        self.energy = 5

    def feed(self) :
        if self.alive :
            self.energy += 3

    def fly(self) :
        if self.alive :
            self.energy -= 1
        if self.energy <= 0 :
            self.alive = False

    def show(self) :
        if self.alive:
            return "-^'"  # That's supposed to be an ASCII hummingbird. Use your␣
↪imagination
        else :
            return "_o___"  # Same hummingbird, deceased (ran out of energy)
```

A `Hummingbird` object has two data attributes, indicating whether it is alive and how much energy it has left. The hummingbird can `feed`, that is, take a sip from a feeder or flower, which makes it gain energy, and `fly`, which consumes energy. These methods modify the data attribute `energy`, but only if the hummingbird is alive. If the hummingbird keeps flying without getting a chance to feed, it will run out of energy and die. The method `show` returns a string with very bad ASCII art representing a live or dead hummingbird.

```
h = Hummingbird()
print(h.show())
h.fly()
print(h.show())
h.feed()
print(h.show())
```

(continues on next page)

(continued from previous page)

```
h.fly()
print(h.show())
h.fly()
print(h.show())
h.fly()
print(h.show())
h.fly()
print(h.show())
h.fly()
print(h.show())
h.fly()
print(h.show())
h.fly()
print(h.show())
```

Apologies to all hummingbird lovers. (Or you can consider this as an encouragement to plant some hummingbird-friendly flowers and keep your feeder full.)

For a better way to visualize mutable objects, consider a class representing a car to be used in a simulation. First we need to import a few pieces from the libraries `matplotlib`, `svgpath2mpl` and `svgpathtools`. You'll get an error when you execute that cell if your Python installation doesn't have those libraries. The section "Other Libraries" below talks through how to install libraries you don't already have.

```
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline
from svgpath2mpl import parse_path
from svgpathtools import svg2paths
```

Details in the class `Car` below are explained in the comments. (This class is based on code used to produce the animation in the notebook "Random Number Simulation of a Drive Thru" in the "Probability Models" chapter. Thanks to Ziling Zong who wrote the original code.)

```
class Car:
    # To make a new car, indicate the coordinates of the car's initial position,
    # the angle (in degrees from the positive x axis) that the car is facing,
    # and the color of the car. Color is a string, for example "red"
    def __init__(self, x, y, degree, color):
        # Set the instance variables. Notice that the first four instance variables
        # have the same names as the parameters
        self.x = x
        self.y = y
        self.degree = degree
        self.color = color
        # The instance variable car_marker is used for the graphical display
        # and requires a little bit of processing
        car_path, attributes = svg2paths("car_outline.svg")
        self.car_marker = parse_path(attributes[0]['d'])
        self.car_marker.vertices -= self.car_marker.vertices.mean(axis=0)

    # Move the car a certain number of pixels in its current direction
    def move(self, amount):
        # Convert the angle the car is facing. We need radians.
        radians = self.degree * math.pi / 180
        # Use trig functions to compute the new x and y coordinates
        self.x += amount * math.cos(radians)
```

(continues on next page)

(continued from previous page)

```
        self.y -= amount * math.sin(radians)

    # Turn the car a certain number of degrees. Positive degrees steers to the
    # left, negative to the right
    def turn(self, amount) :
        self.degree += amount

    # Display an image of the car on a background
    def show_car(self) :
        building = "background.png"
        fig, ax = plt.subplots()
        background = plt.imread(building)
        ax.imshow(background)

        transformed_marker = self.car_marker.transformed(mpl.transforms.Affine2D().
↪rotate_deg(self.degree))
        transformed_marker = transformed_marker.transformed(mpl.transforms.Affine2D().
↪scale(1,1))
        plt.plot(self.x, self.y, marker=transformed_marker, markersize=35, color =␣
↪self.color)
```

Let's make a red car facing due east (0 degrees), starting at point (100, 550).

```
car = Car(100, 550, 0, 'red')
```

The background image shows a drive-thru restaurant with two drive-thru lanes. The point (100, 550) was chosen because that puts the car at the beginning of the outer lane.

```
car.show_car()
```

Let's nudge the car ahead and then see what it looks like.

```
car.move(75)
car.show_car()
```

Now let's steer and move so that the car begins to navigate the lane.

```
car.turn(15)
car.move(75)
car.show_car()
```

It looks like the car is crashing into the drive-thru speaker. Oh well. If you run the previous cell seven more times, you'll see that after running over the speaker the car more or less rounds the east side of the restaurant without hitting anything else.

In a new series of cells, make a new car with a color of your choice and see if you can steer the car safely through the lane using calls to `move` and `turn` (as well as `show_car` to track your progress).

3.2.9 Section on numpy

The NumPy library is a widely-used package for numerical computing in Python. Several other packages for data analysis, scientific computing, machine learning, and the like make use of NumPy.

```
import numpy as np
```

The key part of the NumPy library is the array. If you learned programming in a language like C or Java, you are already familiar with arrays; for the purposes of Python, we can think of them as being like lists except that they make more efficient use of computer memory. Mathematically, they are the natural way to represent vectors and matrices.

Observe by example. There are several ways to make an array. We can make an array from a list.

```
aa = np.array([1,2,4,5,6])
print(type(aa))
print(aa)
```

We can make an "empty" array—or, really, an array of all zeros:

```
bb = np.zeros(10)
print(bb)
```

We can make an array that contains all the values in a range:

```
cc = np.arange(5,15,.75)
print(cc)
```

Let's look carefully at that function above, `arange`. The name come froms "**a**rray **range**"—it's not the English word *arrange*. It is based on the standard `range` function we saw earlier. The three parameters are the (inclusive) start, (exclusive) end, and step size.

Another function to make a new array has a subtle but important difference from `arange`. The `linspace` function takes an inclusive beginning and ending value, but, instead of step size, it takes a number of values. The values themselves are uniformly spread through the range.

```
dd = np.linspace(5, 15, 10)
print(dd)
ee = np.linspace(5, 15, 25)
print(ee)
```

The function's name means "linear space," and it is especially useful in making graphs, since it can make an array to contain the x values at which to plot points. You can increase the graph's resolution by increasing the third parameter to the function.

An important concept with arrays is their *shape*. For all the arrays we have made so far, the shape is just the size or number of elements.

```
print(aa.shape)
print(bb.shape)
print(cc.shape)
print(dd.shape)
print(ee.shape)
```

The reason why the shape is formatted that way is because these arrays have only one dimension—but we can make arrays of any number of dimensions. Here we make a 4-by-6 array of all zeros:

```
mm = np.zeros((4,6))
print(mm.shape)
print(mm)
```

Arrays also can be reshaped.

```
ff = np.arange(0,24)
print(ff)
gg = ff.reshape((2,12))
print(gg)
hh = ff.reshape((6,4))
print(hh)
```

The NumPy library provides a wide range of tools for indexing into an array. First notice that the following give the same result.

```
print(hh[2][3])
print(hh[2,3])
```

The second of these opens the door to various ways to take a slice of a two-dimensional array.

```
# Grab a row
print(hh[1,:])
# Grab a column---although it will be formatted like a row
print(hh[:,1])
# Grab everything but the outer rows and columns
print(hh[1:6,1:4])
```

Just as arrays are the natural representation of vectors, two-dimensional arrays are the natural representation of matrices. NumPy arrays support a full range of vector and matrix operations, many of which can be used through the same symbols as plain old arithmetic. To conclude, think through these examples:

```
# Make two vectors of the same shape
v1 = np.array([1,3,7,9])
v2 = np.array([2,12,8,1])

# We can add two vectors
print(v1 + v2)

# The * symbol does component-wise multiplication
print(v1 * v2)

# The @ symbol does matrix multiplication, but for two
# vectors that amount to the dot product
print(v1 @ v2)

# We can make @ give us the cross product by reshaping
# the vectors into single-column or single-row matrices.
# The result is a matrix
m = v1.reshape((4,1)) @ v2.reshape((1,4))
print(m)

# Mixing arrays and numbers in arithmetic operations
# performs component-wise arithmetic with the given scalar
print(v1 + 5)
print(1/m)
```

3.2.10 Trouble shooting libraries

As you work through code found in other chapters of this book, you may encounter notebooks that use libraries not installed on your system. Running this code will result in an error like `ModuleNotFound`. Depending on your platform or installation, this problem may turn out to be easy to fix. You can search for the missing Python library on the Web. Most widely-used libraries have accompanying websites that include installation instructions, usually involving the `pip` tool. For example, the seaborn library for statistical data visualization is used by several of this book's chapters, and it can be installed by running the following command in commandline interface ("Terminal" on Mac, "Powershell" on MS Windows):

```
pip install seaborn
```

Of course, all of this is just the beginning of Python and its libraries for math, data science, and related fields. The next chapter follows up by showing Python as a tool for working with data.

3.3 Working with Data

3.3.1 Introduction

In this section, we will bring the Python programming part to be more related to data processing. The organization of this section is as follows: First, we will begin with an overview of data, describe the background of processing data, and explore various data types. Then, we will show how the Python programming language is written to prepare the data for analysis, including reading and writing data. Next, we will specifically introduce pandas, a data analysis library in Python, to demonstrate the ways of data processing and data operations for statistics and analysis. Finally, we will guide you through the basics of Matplotlib, a popular Python library for data visualization.

3.3.2 An Overview of Data and Data Analysis

There are several data-related research areas. A database researcher is interested in how data is stored and managed, whereas a data scientist focuses more on the topics such as methods of processing massive data, analyzing data from various sources, interpreting analyzed data, and visualizing the analytical results. Data analysis, in particular, has drawn attention to statisticians, computer scientists, business people, health professionals, or even philosophers and artists.

Traditionally, data analysis was thought to be done by mathematicians only. Scholars often considered it in the domain of statistics. Today, however, new challenges motivate us to think of data analysis as a broad, multidisciplinary area. We use a variety of tools including computer programs to perform data operations. Due to the scope of this chapter and this book, we describe only a few facts of data and its related challenges in the era of modern data analysis.

At present, scholars are dealing with large-scale data everywhere. For example, the Internet today has billions of webpages. According to the conservative estimates, there are at least three hundred billion emails that are sent daily. The need of analyzing such a great amount of data effectively has driven scientists to develop well-designed methods and to constantly update the hardware to increase the computing power.

The modern world generates data at a fast pace. Even if the data is from the same source, numbers that are collected in the past can be very different from them at present. At the early stage of the COVID-19 pandemic, researchers and decision makers realized that the reported data of disease infections fell behind. The collected data for the daily reporting largely reflected what had happened in the past rather than what was happening.

Data analysts are presented with structured, semi-structured, or unstructured data. One example of a structured dataset may be from a professor's grade book. Students' names and grades of all assignments are presented completely and accurately in a well-organized and cross-checked spreadsheet/table. Less or no additional effort of data "re-structuring" needs to be made. However, data from an ordinary person's diary can be unstructured. The personal note has various lengths, covers different and unrelated topics, and is likely handwritten. Extracting data for an analytical process requires a tremendous amount of data pre-processing work.

Just like water is an important resource for people, agriculture, and industry, data is everywhere and fundamental in many different contexts. However, just like water, the origin and the quality of data matter. The prevalence of data does not imply that all kinds of data can be used without scrutiny. An online survey result that was collected anonymously and assembled by a controversial business entity may be a less reliable and more biased source for data analytics. The clinical trial data submitted by responsible medical professionals and peer reviewed by a reputable research group is thought to be more trustworthy for academic purposes. The extent of data pre-processing depends on the source and the collection method of data.

There are various features of data due to the difference of dataset size, generation method, structure, and its sources. For analyzing data, several methods and challenges exist, too. The direction of data analysis can be predictive or descriptive:

For *predictive* data analysis, one uses existing data to predict unknown and future data. For example, a credit card company can analyze the history of a user's credit card use to detect potential fraud and possibly unauthorized transactions. An economist may analyze data from multiple sources to model the likelihood of a recession in the next few weeks or months. A climate scientist is able to create a model to predict the temperature trend from a region based on the historical pattern of the climate data.

For *descriptive* data analysis, one performs analysis on available data in order to discover and better understand the features of the dataset. For example, a genealogist may use a program to analyze the people's faces to tell how two individuals look similar to or different from each other. A business sales team may perform a market basket analysis from millions of transactions to discover the sets of frequently brought items.

The choice of a data analysis method depends on not only the objective of analysis but also the type of data. Generally speaking, there are four common types of data [`S.S.Stevens`]. *Nominal* data is the names or values themselves, such as postal codes or identification numbers. *Ordinal* data affords a natural ordering or ranking, such as wine's quality grades or customer's satisfaction ratings. *Interval* data also can be ordered, but in this case the difference of data values is more explainable—calendar dates, for example. *Ratio* data can be meaningfully multiplied or divided, such as the distance between two locations.

Data operations must be suitable for the specific data types. We can say that iron is about 7.8 times as heavy as water (the density of iron is 7.87 g/m^3; the density of water is 1 g/m^3). However, it is wrong to do a simple arithmetic sum operation by saying the density of a mixture of iron and water is 8.8 (as 7.8+1) times as heavy as water. In biology, a DNA molecule is composed of four types of bases: adenine (A), thymine (T), guanine (G), and cytosine (C). These four bases are paired as A to T and G to C. However, we cannot just "compare" them to determine which base is "larger" or "precedent" of another, in the usual way that we do with natural numbers.

In the next part of this chapter, we will cover data analysis topics from executing common statistical tasks to building specific models for further analysis. We will demonstrate to the readers how to write Python programs relevant to the theoretical aspect of analysis. Moreover, we will discuss the benefits of utilizing computer programs when the data is processed with the help of Python libraries.

*: Stevens, S. S. (1946). On the theory of scales of measurement. Science, 103(2684), 677-680.

3.3.3 Data Operations: Importing and Exporting, Reading and Writing Data

For the complete process of data analysis, programmers usually follow the sequence of:

1. Reading data from a file or from the user input,
2. Processing loaded data, including re-organizing data, computations, and generating results,
3. Outputting data to the computer screen or writing the output to files.

The first step is pre-processing the data, that is, to prepare the "clean" data for analysis from the raw data. This step usually involves the loading of a dataset that is imported from databases, files, or the user inputs. The raw data can take various forms. As we have mentioned in the previous section, the data can be more or less structured. In this chapter, we will focus on the processing of structured datasets.

A structured dataset, according to this book's definition and its applications, is the one that is:

1. Linear, such as a collection of records with the same type, i.e. 1-dimensional data, or
2. Table-like, such as the records organized with complete information of rows and columns, i.e. 2-dimensional data, or
3. Cuboid-like, such as data points in a space, represented by three values, x, y, z, i.e. 3-dimensional data).

Since the computer screen is "flat", i.e. 2-dimensional, and all data must be properly structured and saved into the memory space, we still need to convert the 3-dimensional data in a form that fits into a 2-dimensional data's view. There are several techniques that do such kind of 3D-to-2D conversion, but this topic is currently beyond the scope of our book. Therefore, we only introduce the methods of processing a file that is already saved with one dimension or two dimensions.

One-dimensional data processing

For a one-dimensional dataset, a Python program can read the data from the user input one at a time. The program may save all user-inputted data into a data structure such as a list.

```
##########------uncomment this block if working with an interactive JNB-------------#
 ↪###############

## The program asks the number of inputs by the user
# print("How many numbers to input")

# num_inputs = int(input()) # The input() function takes a string by default.␣
 ↪Converting to int type.
# data_list = []   # Initial list is empty

## The user inputs records one after one. All will be saved into data_list
# print("Enter each record. Press Enter to continue.")
# for i in range(num_inputs):
#    user_data = input()
#    data_list.append(user_data)

##########-------------------------------------------------------------------------#
 ↪################

#delete the following line if running an interactive JNB
data_list=[4,6,8,2,21,9,5]


# Output the list that contains all previously inputted data
print(data_list)
```

```
[4, 6, 8, 2, 21, 9, 5]
```

Another situation to read the data is from a file. For the convenience of processing, we assume that such a file is a plain text file and it only contains relevant data for calculations (i.e. no data labels, titles, informational fields, or any empty spaces or lines). Furthermore, we assume that all records are stored in one line where the two records are separated by a single white space.

```
# First, open the file that contains all records; store the content to the object␣
 ↪data_file_1.
data_file_1 = open("datafile1.txt", "r")

# Then, read the data_file object for the first (also the only and the last) line and␣
 ↪save it to a string.
one_line = data_file_1.readline()
```

(continues on next page)

(continued from previous page)

```
# Next, split the string into several records and further save the records into a␣
 ↪list.
# Note: the white space is the separator in this program.
data_list = one_line.split()

# Finally, output the data from the saved list, data_list.
print(data_list)

# Close the file.
data_file_1.close()
```

```
['7', '12', '41', '3', '19', '24', '81']
```

Explanation:

The file `open()` function in this code example takes two arguments. The first one is the file name. The second one is the file processing mode. "r" here means read. The `readline()` function outputs only one line of data from a file if it is used once.

The `split()` function is used to split data based on the separator. If nothing is specified in the function's argument, the default separator is white spaces.

As a good programming practice, programmers should close the file before the end of the program by explicitly writing the file closing function `close()`.

On the other hand, we can assume that the data are stored in multiple lines where each line contains one record.

```
# First, open the file that contains all records; store the content to the object␣
 ↪data_file_2.
data_file_2 = open("datafile2.txt", "r")

# Next, use a loop to iterate all lines, where each line has one record.
for line in data_file_2.readlines():
    # Output data
    print(line[:-1]) # -1 is used for not printing the end-of-line's line break␣
 ↪character

# Close the file.
data_file_2.close()
```

```
7
12
41
3
19
24
81
```

Explanation:

The `readlines()` function differs from the `readline()` function. The program will continue to read another line till the end of the file. The for loop is used to iterate each line. The `print()` function inside the for loop will print each line's content except the line break character at the end of each line.

This program treats each line as a string. The code `line[:-1]` means that the program does slicing of the string so that the new string contains all characters from the beginning to the second-to-last character, cutting off the last. The value `-1` used as an index means the last position, and `:-1` indicates that the last position is an exclusive upper bound. When

reading a plain-text file, the last character on a line is the line break itself, and so `line[:-1]` simply gets rid of the line break character.

In most programming languages, including Python, such a line break character is represented by `\n`. That `\n` is not a part of data itself and thus should be removed. Please note that we must ensure that the last record or line of the file is also followed by a line break (`\n`) or that record will not be processed correctly.

Two-dimensional data processing

An efficient way of storing the data in a table (i.e. two-dimensional data) is using a file format called Comma-Separated Values (CSV). Such a file is named like `birds.csv`. A typical `.csv` file looks like the following:

```
# Import the pandas library. Here we rename "pandas" as "pd" for convenience.
import pandas as pd

# Read and load the .csv data from the specified file location into the program's␣
↪memory space,
# and save the data into a format called "DataFrame".
my_data = pd.read_csv('birds.csv')

# Operate on the object with the DataFrame format.
print(my_data)
```

```
                      Name  Weight  Clutch size  Wingspan     Family (common)  \
0        American goldfinch    15.5            5      20.5             finches
1            American robin    83.0            4      36.5            thrushes
2   black-capped chickadee    11.5            7      18.5                tits
3                 blue jay    85.0            4      38.5             corvids
4          eastern bluebird    30.0            4      28.5            thrushes
5             mourning dove   141.0            2      41.5   doves and pigeons
6         northern cardinal    45.0            4      28.5           cardinals
7         redwing blackbird    54.5            4      14.5          blackbirds
8          Baltimore oriole    35.0            4      27.5          blackbirds
9           dark-eyed junco    24.0            4      21.5  new-world sparrows
10          common starling    78.0            6      37.5           starlings

   Family (scientific)
0         Fringillidae
1             Turdidae
2              Paridae
3             Corvidae
4             Turdidae
5           Columbidae
6         Cardinalidae
7            Icteridae
8            Icteridae
9        Passerellidae
10           Sturnidae
```

Explanation:

On line 1, we import the Pandas library, but we may not want to write the complete name of pandas every time when we use it. Therefore, we use the alias `pd` to denote `pandas`.

On line 2, `my_data` is the object with the DataFrame format. It is used to be processed by the Pandas library according to the row and column information. `pd.read_csv` is the function from the Pandas library to read the `.csv` file. Enclosed in the function's parameter 'data.csv' is the file name (data.csv).

If your Python's workspace and the data file are in the same directory/folder, no further action is needed from this part. However, if the data file is stored elsewhere, we must specify the location of such a file in the Python code, such as:

```
my_data2 = pd.read_csv('user/math_teacher/Desktop/project2/data2.csv')
```

or

```
my_data3 = pd.read_csv('C:\\Users\\Math_Student\\Desktop\\Project3\\data3.
csv')
```

Note that the Python code for specifying a file location depends on the operating system the user is using. The first form is usually suitable for MacOS and Linux systems, where a forward slash (/) is used to go into a sub-directory. The second form is often seen in a system where a backward slash (\) is used to go into a sub-directory, such as Microsoft Windows. Here, two backward slashes are used because Python processes the special character \ with an escape character . The first \ is for the formatting purpose to indicate that there is a special character coming next and the second \ is the actual character to be read.

As the file name has suggested, the file contains comma-separated values because commas (,) are commonly used to separate the columns. However, files can still be processed by the Python program if the columns are separated by a symbol other than a comma. Such a symbol may be a white space ( ), a semi-colon (;), or any other symbol.

In order to read such kind of file and process the column information correctly, we change the parameter part "sep=X" to indicate a different separator. "X" is replaced by the separator.

For example,

```
my_data4 = pd.read_csv('data4.csv', sep = ' ')
```

or

```
my_data5 = pd.read_csv('data5.csv', sep = ';')
```

From the above examples, the first one is used to process a file where columns are separated by a single white space. The second one is by a semi-colon. If no parameter is specified, this read_csv function will process the file that treats the comma as the separator by default.

Exporting Data: Writing Data to A File

Next, we describe the ways to export and present data. If the output is to the computer screen, Python's `print()` function should have the need. If we want to output data to a file, we will use Python's file writing functions. The following code can be used to write the data to a file. At this time, we also assume that the file is in a plain text format, which is more conventional and convenient for the data storage, other than a binary format, which is encoded for the use by a computer program and/or program instructions.

```
# Assuming that all generated data have been stored in a list….
generated_data = [10, 20, 30, 4, 5, 6]

# First, open the file to which the data is being written to
output_file = open("results.txt", "w")

# Next, iterate all elements in the list so that each element is written to the same␣
↪file
for i in generated_data:
    output_file.write(str(i) + " ") # elements one after one, separated by a white␣
↪space.

# Close the file.
output_file.close()
```

Explanation:

In this example, we assume that all data have already been stored in a list. Since the file operation is to write to a file, we use the parameter "w" within the `open()` function to write data to a file. results.txt is the plain text file name that all data is written to.

The program will process the for loop by accessing each element in the list one by one. In each iteration of the loop, the program executes the write() function by pushing one element into the file. The file writing function, `write()`, always needs to process strings only (the system treats the output as a long string) but the user generated data may not be strings. Just like in our example, we must convert the number type data, e.g. int, into a string type. Therefore, we use `str()` to do such a type conversion.

After one element is written into the file, a white space is also written to the file before the loop repeats to write another element into the file. Finally, the program closes the file with the `close()` function.

The code example works to output all data into a file, where all records are stored in one line only. What if the user intends to output data with multiple lines, where each line contains one record only? What if the user decides to save the output file in a directory other than the Python program's default working directory? We are leaving these questions to the readers.

3.3.4 Introduction to Pandas

The code in the previous section used the `pandas` library to read csv data. But `pandas` does a lot more, and in fact is one of researchers' top selections of libraries for its convenience and effectiveness in performing data analysis. According to the official website, `pandas` "*is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.*"(https://pandas.pydata.org/) The letters from the library's name appear to be taken from "**P**ython **D**ata **ANA**ly**S**is" by its creator.

`pandas` can be used to process datasets with various dimensions and structures. They include a one-dimensional dataset – processed as `Series`, a two-dimensional dataset – processed as `Dataframe`, a traditionally more structured dataset such as a .csv file, or a more complex but less structured dataset, such as a .json file. Users who are familiar with other programming languages and statistical applications, including R, SAS, Stata, SQL, and spreadsheets will find similar tools and operations of processing data in `pandas`.

Users in `pandas` can structure the data to be compatible with a spreadsheet software application (e.g. Microsoft Excel, LibreOffice Calc, Apple Numbers). Here are the important `pandas` data structures and their equivalent names in a spreadsheet application:

- Series – column
- Dataframe – entire worksheet
- Index – row headings
- NaN – empty cell

As we see, Panda's `Series` is used to store and process a list of elements, i.e. data from one column. `Dataframe` is suitable to process data from a table. In the next section, we will explain the data operations associated with `pandas Series` and `Dataframe`.

Data operations with `pandas.Series`

The Pandas library has its built-in functions to process one-dimensional data and two-dimensional data. To perform a variety of tasks with a one-dimensional dataset, we use the functions from `pandas.Series`. The following example shows how we load a list of elements into the Series.

```
import pandas as pd
# Original data are stored in a list.
data_list = [100, 200, 300, 123, 654, 99, 98, 100, 45, 100, 101]
# New data to be stored in a Series, namely data_series
data_series = pd.Series(data_list)
# Print elements from Series.
print(data_series)
```

```
0     100
1     200
2     300
3     123
4     654
5      99
6      98
7     100
8      45
9     100
10    101
dtype: int64
```

Notice the output from the `print()` function. Values from the first column are viewed as the Series index numbers, which start from 0. The second column contains the actual records from the Series data. Because the list is a one-dimensional dataset, the index value has no additional meaning other than telling the user the position (by zero-based indexing) of an element. While it is certainly possible to operate on data using Python's list structure, including writing codes and calculating numbers, Panda's `Series` has an advantage as it has provided many commonly used functions for mathematical purposes.

Following the previously made code, we can use available tools from `pandas.Series` to quickly get useful statistical results.

```
# Getting the maximum value from the series
data_series.max()
```

```
654
```

```
# Getting the minimum value from the series
data_series.min()
```

```
45
```

```
# Getting the mean value from the series
data_series.mean()
```

```
174.54545454545453
```

```
# Getting the median value from the series
data_series.median()
```

```
100.0
```

```
# Getting the mode(s) of the Series.
# A series may have multiple modes, i.e. more than one value appears the most often.
data_series.mode()
```

```
0    100
dtype: int64
```

```
# Getting the nth largest values from the series, using .largest(n).
n = 5
data_series.nlargest(n)
```

```
4     654
2     300
1     200
3     123
10    101
dtype: int64
```

```
# Getting the nth smallest values from the series, using .smallest(n).
n = 4
data_series.nsmallest(n)
```

```
8     45
6     98
5     99
0    100
dtype: int64
```

```
# Getting the standard deviation from the series.
# Such a standard deviation is normalized by N-1 by default.
data_series.std()
```

```
173.0169723676632
```

```
# Getting the variance from the series
data_series.var()
```

```
29934.87272727273
```

```
# Getting a quick statistical summary of data from the series
data_series.describe()
```

```
count     11.000000
mean     174.545455
std      173.016972
min       45.000000
25%       99.500000
50%      100.000000
75%      161.500000
max      654.000000
dtype: float64
```

```
# Getting the unique elements from the series
data_series.unique()
```

```
array([100, 200, 300, 123, 654,  99,  98,  45, 101])
```

```
# Getting the number count of unique elements from the series
data_series.nunique()
```

```
9
```

```
# Getting the size of its series, i.e. how many elements in this series
data_series.size
```

```
11
```

Note that when programmers write the code to get the series' size, they write `.size` instead of `.size()`. This is because `size` is a data attribute (or instance variable) of the `Seires` class. Recall from the introduction to classes and objects in the previous chapter that classes have both data attributes and function attributes (or methods).

Attributes such as `max()` and `min()` are defined as methods because they are actions that are performed on data. An action has its internal processing procedure and may have an output following its processing. For example, `max()` has its internal processing procedure that compares the values of all records and selects the maximum value. Then, it outputs the selected maximum value. Thus, we will use the output number from the `max()` function.

Data operations with pandas.Dataframe

If the dataset is two-dimensional, we rely on Pandas's `Dataframe` to process it. We have explained the use of Pandas library to read and extract information from a `.csv` file in the previous section of the chapter. Now, we will demonstrate how to use `Dataframe` to select and operate on the two-dimensional data.

To demonstrate the operations with panda's Dataframe, we use the dataset from `Car Evaluation Database` that is downloadable from UCI Machine Learning Repository (https://archive.ics.uci.edu/dataset/19/car+evaluation). First, we download the .zip package that contains the dataset file "car.data". Next, we change the file name from "car.data" to "car.csv". This step is optional in the real-world data analysis since the file has already been perfectly formatted with comma-separated values. We do the file name change only for consistency in our explanation. Then we modify the file by adding a title line for our data processing need:

buying,maint,doors,persons,lug_boot,safety,class

This step can be done manually by opening the data file or by writing a line of Python code, which we will demonstrate shortly.

This dataset was originally used for predicting a car's acceptability level based on several conditions of that car. Each row shows the conditions (attributes) and acceptability level (class value). As we look into the dataset, the `Car Evaluation Database` contains the following information:

Class values:

- unacc, acc, good, vgood

Explanation:

- class (acceptability level: unacceptable, acceptable, good, and very good).

Attributes:

- buying: vhigh, high, med, low.

- maint: vhigh, high, med, low.
- doors: 2, 3, 4, 5more.
- persons: 2, 4, more.
- lug_boot: small, med, big.
- safety: low, med, high.

Explanation:

- buying (buying price: very high, high, medium, and low),
- maintenance (price of maintenance: very high, high, medium, and low),
- doors (number of doors: 2, 3, 4, and 5 or more),
- persons (capacity in terms of persons to carry: 2, 4, and more),
- lug_boot (the size of luggage boot: small, medium, and big), and
- safety (estimated safety of the car: low, medium, and high).

```
# Use of Dataframe in pandas

import pandas as pd
#   Loading the dataset to let pandas read the .csv file.
loaded_data = pd.read_csv('car.csv')  # or loaded_data = pd.read_csv('car.data') for␣
 ↪the original file name

# data_car has its title added. All column names are specified in this step.
data_car = pd.DataFrame(loaded_data, columns = ['buying','maint','doors','persons',
 ↪'lug_boot','safety','class'])

# Displaying the overview of the dataset
print(data_car)

# Selecting and displaying the data from the column 'buying'
print(data_car['buying'])
```

```
     buying  maint  doors persons lug_boot safety  class
0     vhigh  vhigh      2       2    small    low  unacc
1     vhigh  vhigh      2       2    small    med  unacc
2     vhigh  vhigh      2       2    small   high  unacc
3     vhigh  vhigh      2       2      med    low  unacc
4     vhigh  vhigh      2       2      med    med  unacc
...     ...    ...    ...     ...      ...    ...    ...
1723    low    low  5more    more      med    med   good
1724    low    low  5more    more      med   high  vgood
1725    low    low  5more    more      big    low  unacc
1726    low    low  5more    more      big    med   good
1727    low    low  5more    more      big   high  vgood

[1728 rows x 7 columns]
0       vhigh
1       vhigh
2       vhigh
3       vhigh
4       vhigh
        ...
1723      low
```

(continues on next page)

(continued from previous page)

```
1724      low
1725      low
1726      low
1727      low
Name: buying, Length: 1728, dtype: object
```

One can select specific row(s) and columns from the Dataframe by using `loc` (to be understood as "location") or `iloc` (to be understood as "index-location") subscripts. `loc` is used when a label is present for rows and columns. `iloc` is used when the selection is based on the index values of rows and columns.

To use iloc, the subscript [ ] takes the following format: [R : C], where R is the row index and C is the column index, or [R1:R2, C1:C2], where R1 is the starting row, R2 is the ending row minus one, C1 is the starting column and C2 is the ending column minus one. If any variable, R1, R2, C1, or C2 is left blank, the selection is by default from the first row or column to the last row or column. Note that the row information and structure (the colon ":") must be present, but the column part is optional.

Here are examples of making selections of rows and columns.

```
# Selecting one row, row index 5 from the dataset
data_car.iloc[5, ]
```

```
buying      vhigh
maint       vhigh
doors           2
persons         2
lug_boot      med
safety       high
class       unacc
Name: 5, dtype: object
```

```
# Selecting 3 rows from index 7 to index 9 (i.e. 10 minus 1) from the dataset
data_car.iloc[7:10, ]
```

```
  buying  maint doors persons lug_boot safety  class
7  vhigh  vhigh     2       2      big    med  unacc
8  vhigh  vhigh     2       2      big   high  unacc
9  vhigh  vhigh     2       4    small    low  unacc
```

```
# Selecting one column, column index 2 from the dataset
data_car.iloc[ : , 2]
```

```
0           2
1           2
2           2
3           2
4           2
        ...
1723    5more
1724    5more
1725    5more
1726    5more
1727    5more
Name: doors, Length: 1728, dtype: object
```

```
# Selecting 2 columns from index 2 to index 4 (i.e. 5 minus 1) from the dataset
data_car.iloc[ : , 2:5]
```

```
      doors persons lug_boot
0         2       2    small
1         2       2    small
2         2       2    small
3         2       2      med
4         2       2      med
...     ...     ...      ...
1723  5more    more      med
1724  5more    more      med
1725  5more    more      big
1726  5more    more      big
1727  5more    more      big

[1728 rows x 3 columns]
```

```
# Selecting 4 rows from index 4 to index 7 and 3 columns from index 1 to index 3 from␣
 ↪the dataset
data_car.iloc[4:8, 1:4]
```

```
   maint doors persons
4  vhigh     2       2
5  vhigh     2       2
6  vhigh     2       2
7  vhigh     2       2
```

If one does not want to select continuous rows or columns, a Python loop may be applied to meet the selection need. Take a look at the following example.

```
# Iterating through each row; selecting index column 0 and 2 respectively.
for i in range(10):
  print(data_car .iloc[i, 0], data_car.iloc[i, 2])
```

```
vhigh 2
vhigh 2
vhigh 2
vhigh 2
vhigh 2
vhigh 2
vhigh 2
vhigh 2
vhigh 2
vhigh 2
```

```
# Using value_counts() to get counts of attributes
data_car['buying'].value_counts()
```

```
med      432
high     432
low      432
vhigh    432
Name: buying, dtype: int64
```

```
# Getting the unique elements from a column, e.g. 'lug_boot', using .unique()
data_car['lug_boot'].unique()
```

```
array(['small', 'med', 'big'], dtype=object)
```

```
# Getting the number count of unique elements from a column, e.g. 'lug_boot', using␣
↪nunique()
data_car['lug_boot'].nunique()
```

```
3
```

```
# Getting the number count of elements for each column
data_car.count()
```

```
buying      1728
maint       1728
doors       1728
persons     1728
lug_boot    1728
safety      1728
class       1728
dtype: int64
```

```
# Getting the total number count of elements from all columns (i.e. the entire␣
↪dataframe)
data_car.size
```

```
12096
```

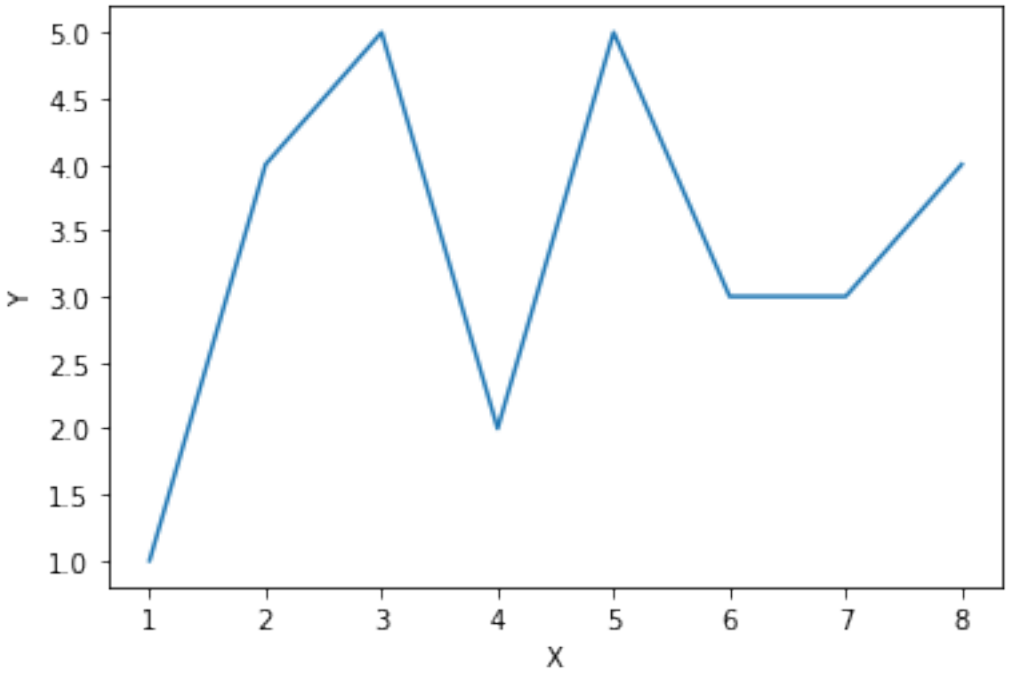
3.3.5 Basics of matplotlib

There is an old saying, a picture is worth a thousand words. Data visualization enables a reader to perceive and process information in a visualized form. Technically, data visualization transforms numerical data into a visual form. It facilitates both professionals and the general public to understand complex analytical results. Moreover, it is an important and interesting topic for data analysis.

There are several dedicated Python libraries for data visualization, including Matplotlib, Plotly, and Altair. In this section, we will talk about data visualization in general and describe the basic use of Matplotlib for plotting charts. More information about the use of Matplotlib can be found from its official website: https://matplotlib.org/

In this chapter, we will focus on the creation, description, and comparison of two-dimensional charts. Let's take the first example of visualization with Matplotlib. The following code creates a line chart from the (x,y) pairs of data.

```
# Example 1: A line chart
# import matplotlib.pyplot for visualization
import matplotlib.pyplot as plt
# data for pairs x and y
x_values = [1, 2, 3, 4, 5, 6, 7, 8]
y_values = [1, 4, 5, 2, 5, 3, 3, 4]
# plotting and labeling
plt.plot(x_values, y_values)
plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```

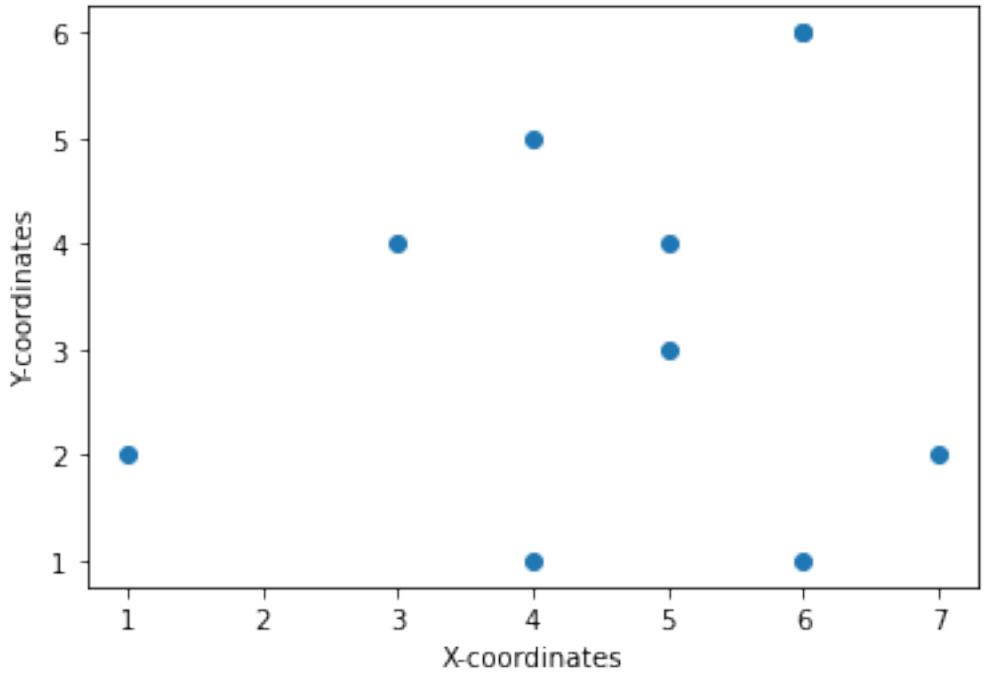


Explanation: matplotlib is a giant package that includes lots of visualization sub-packages and functions. We may specify the collection of functions that will be used in plotting charts, such as line charts, pie charts, etc. Therefore, we import `matplotlib.pyplot` instead of `matplotlib` so that we do not have to repeatedly write pyplot when we use its functions. Since matplotlib.pyplot is a relatively long name, we conventionally use the shorter name plt to substitute it for convenience.

The `plot90` function does the visualization, and `xlabel` and `ylabel` are used for the naming of x-axis and y-axis. Finally, we use the `show()` function to let the chart display in the Jupyter Notebook.

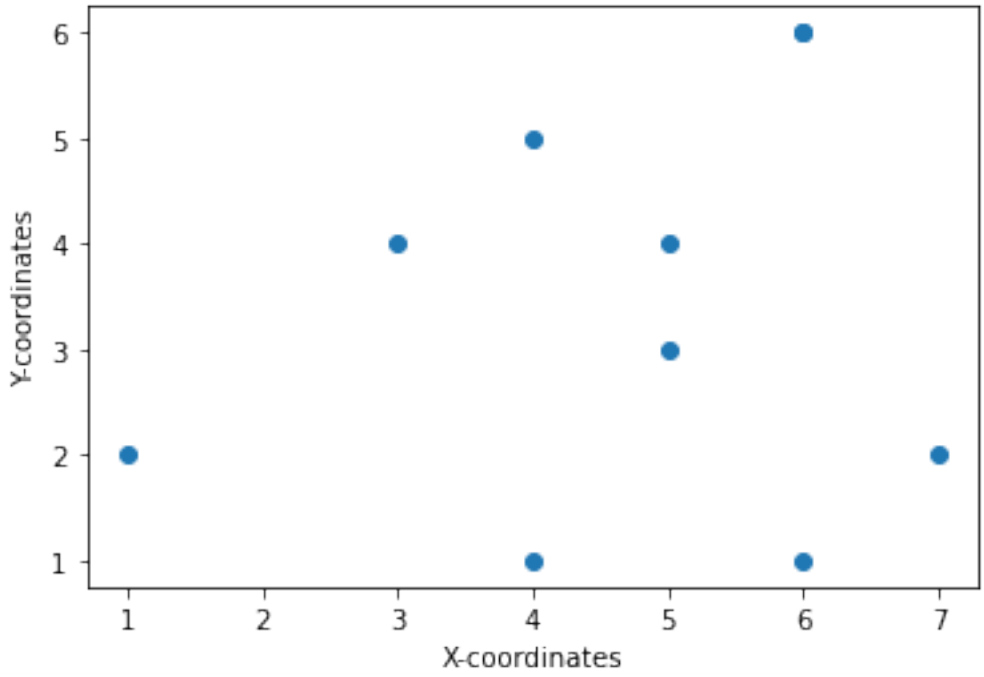
In a line chart, data points are connected by lines. We generally use a line chart when there are continuous data, e.g. stock price changes, heart rate monitoring. Occasionally, data points should not be connected as they are independent values, such as location coordinates. A scatter plot will be suitable for this kind of display.

```
# Example 2: a scatter plot
# import matplotlib.pyplot for visualization
import matplotlib.pyplot as plt
# Integrate matplotlib displays with Jupyter
%matplotlib inline


# data for pairs x and y
x_points = [1, 3, 4, 4, 5, 5, 6, 6, 6, 7]
y_points = [2, 4, 1, 5, 4, 3, 6, 6, 1, 2]
# plotting and labeling
plt.scatter(x_points, y_points)
plt.xlabel('X-coordinates')
plt.ylabel('Y-coordinates')
plt.show()
```



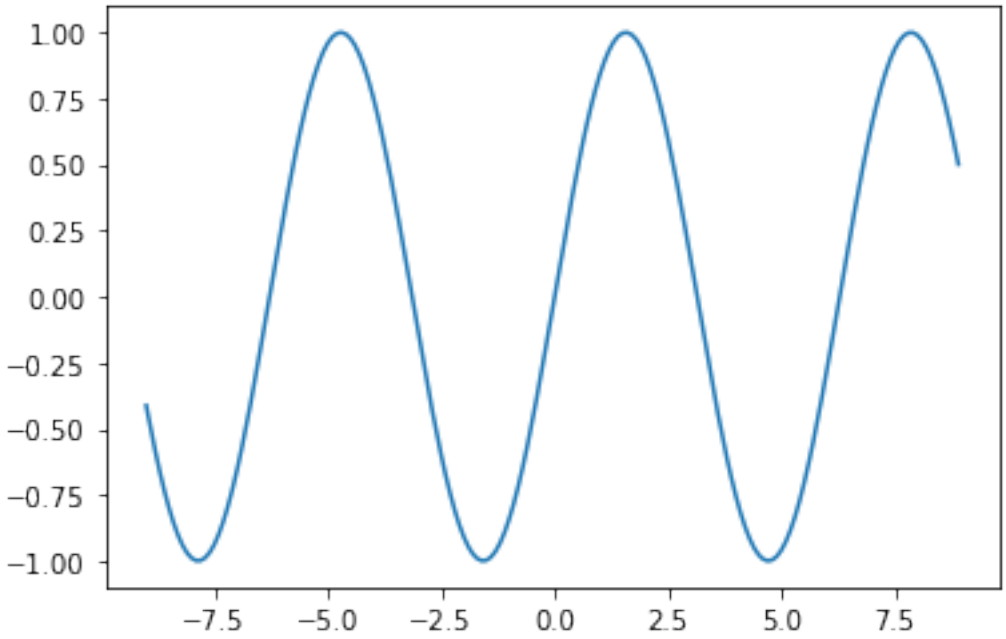
Sometimes, users prefer implementing charts using arrays from the Numpy library (see the previous chapter) instead of lists. We can re-write the above code to use Numpy arrays.

```
# Example 2.1 : a scatter plot, data from numpy
# import matplotlib.pyplot for visualization, numpy for arrays of data
import matplotlib.pyplot as plt
import numpy as np
# data for pairs x and y
x_points = np.array([1, 3, 4, 4, 5, 5, 6, 6, 6, 7])
y_points = np.array([2, 4, 1, 5, 4, 3, 6, 6, 1, 2])
# plotting and labeling
plt.scatter(x_points, y_points)
plt.xlabel('X-coordinates')
plt.ylabel('Y-coordinates')
plt.show()
```



Explanation: Just as we deal with `matplotlib.pyplot`, we conventionally take `numpy` as `np` for convenience. All data from `x_points` and `y_points` are incorporated into Numpy's array structures. With `np.array`, all data points are stored and processed in a similar manner as with Python's basic arrays.

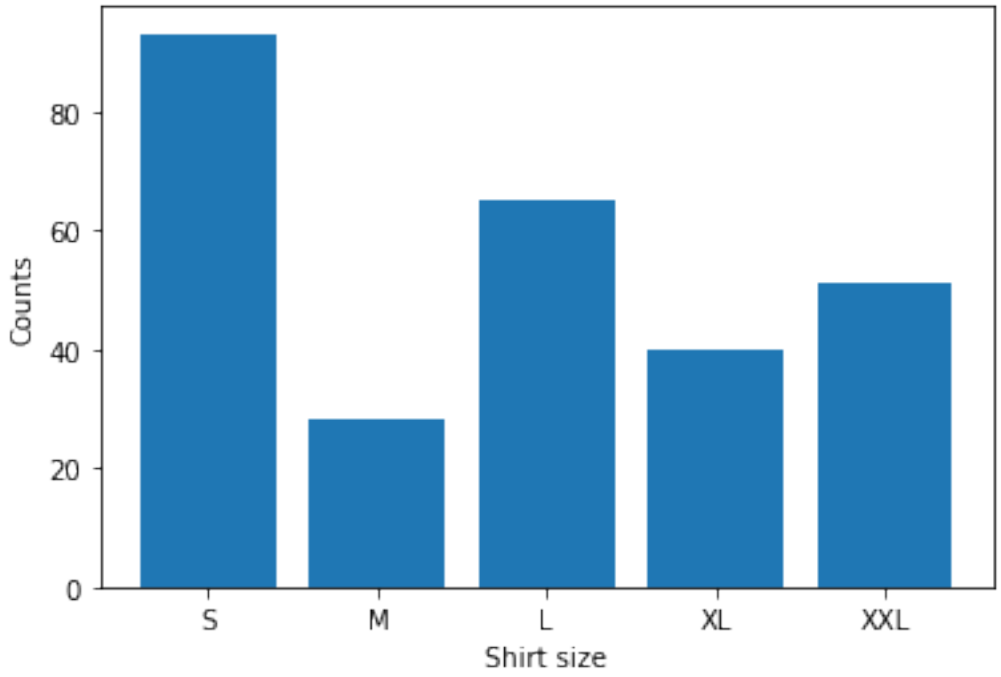
```
# Example 3: plotting a sine wave
# import matplotlib.pyplot and numpy
import matplotlib.pyplot as plt
import numpy as np
# Specifying the range of x, starting value, ending value, and intervals
x_range = np.arange(-9, 9, 0.1)
# Applying the sine wave
y = np.sin(x_range)
plt.plot(x_range, y)
plt.show()
```



Explanation: From the above code, numpy's `arange()` function takes three parameters. The first two determine the range of x values for displaying the sine wave. The third parameter is the intervals between two data points. A sine function itself treats and outputs continuous data but our Python's code still deals with the "dots" of data with numpy. Therefore, we make the plot look like a curved data by placing a sufficient number of data points. The curve will look smooth if more points are added or rough if fewer are there. An interval of 0.1 as shown in the example means that the there is a distance of x=0.1 between two data points. x_range can be viewed as time and y can be viewed as amplitude.

Another useful chart in visualization is the bar chart. Data from different categories may be presented with a bar chart. For example, a business would like to display the count of different shirt sizes from the its inventory. The following code does its job by a bar chart.

```
# Example 4: plotting a bar chart
# import matplotlib.pyplot and numpy
import matplotlib.pyplot as plt
import numpy as np
# Specifying x as catagories and y as value counts for its related catagory
x_cat = np.array(["S", "M", "L", "XL", "XXL"])
y_counts = np.array([93, 28, 65, 40, 51])
# Displaying the bar chart with labels
plt.bar(x_cat, y_counts)
plt.xlabel("Shirt size")
plt.ylabel("Counts")
plt.show()
```



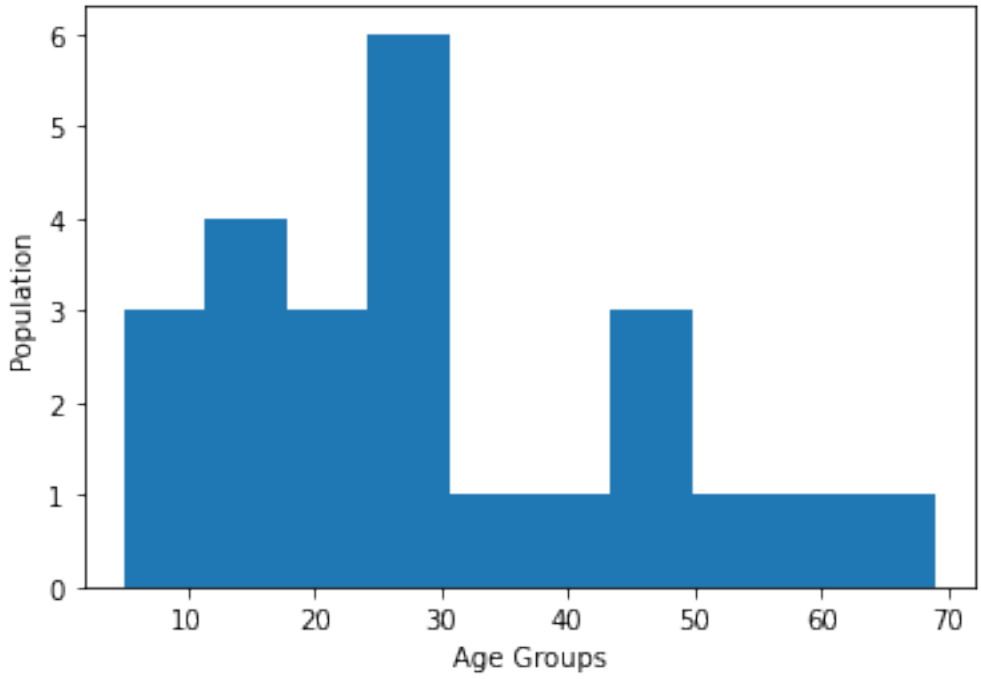
If one needs to present data with a horizontal bar, we can simply replace the `.bar()` function with `.barh()`, as `plt.barh(x_cat, y_counts)`.

Both the bar chart and the line chart are used to represent quantitative data. Both may be suitable to display continuous data. However, a bar chart should be chosen if such data is discrete data.

Similar to a bar chart, a histogram is used for displaying the distribution of data frequencies. Although a bar chart and a histogram look similar and both are used in statistical representation of data, there are differences. In a histogram, bars are next to each other without any interval. A histogram is a graphical representation of quantitative data, such as non-discrete data, whereas a bar chart is of categorical data.

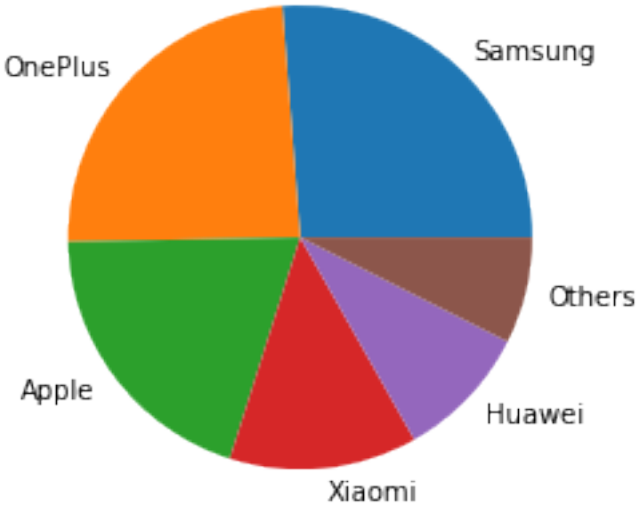
We demonstrate a histogram with the following example. It is for displaying the population among age groups in an area.

```
# Example 5: plotting a histogram
# import matplotlib.pyplot and numpy
import matplotlib.pyplot as plt
import numpy as np
# Specifying ages
ages = np.array([14, 30, 20, 31, 27, 25, 21, 9, 30, 45, 25, 58, 17, 69, 16, 40, 8, 13,
 ↪ 44, 51, 27, 49, 19, 5])
# Displaying the histogram with labels
plt.hist(ages)
plt.xlabel("Age Groups")
plt.ylabel("Population")
plt.show()
```



What if a user wants to present the data that will better display the ratio of each of the categories instead of the absolute value of them? A pie chart may be a good choice. Data values presented in the example are purely fictional and do not reflect any market share in any region in the real world.

```
# Example 6: plotting a pie chart
# import matplotlib.pyplot and numpy
import matplotlib.pyplot as plt
import numpy as np
# Specifying shares of the brands and the names of each associated brand.
brand_shares = np.array([2014, 1848, 1520, 1005, 722, 569] )
brand_labels = ["Samsung", "OnePlus", "Apple", "Xiaomi", "Huawei", "Others"]
# Displaying the pie chart with labels attached to each share.
plt.pie(brand_shares, labels = brand_labels)
plt.show()
```

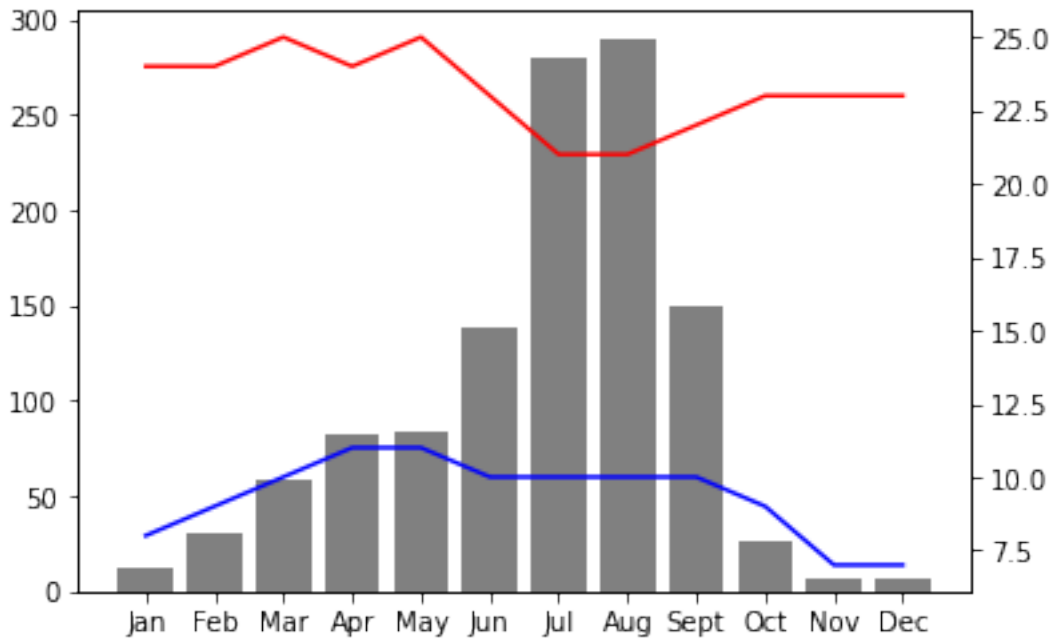


Now, we put the above-mentioned information together to make a graphical representation of data in the real world. For example, we may be interested in a climate data visualization. One would find it easy to read the temperature changes and precipitation differences through charts instead of a string of numbers. In particular, we visualize the monthly climate data for Addis Ababa, Ethiopia. The related data source can be found from the World Meteorological Organization.

In our approach, we present the precipitation data with a bar chart and the temperature data with a line chart using the `matplotlib` library. The precipitation data is according to the average monthly rainfall in millimeters. The temperature data is according to the mean daily minimum temperature and the mean daily maximum temperature in Celsius (°C). The readings are monthly, from January to December.

- Precipitation: 13, 30, 58, 82, 84, 138, 280, 290, 149, 27, 7, 7
- Daily high temperature: 24, 24, 25, 24, 25, 23, 21, 21, 22, 23, 23, 23
- Daily low temperature: 8, 9, 10, 11, 11, 10, 10, 10, 10, 9, 7, 7

```
# Import the library
import matplotlib.pyplot as plt

months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sept', 'Oct', 'Nov
 ↪', 'Dec']

# Store the data in Python lists
precipitation = [13, 30, 58, 82, 84, 138, 280, 290, 149, 27, 7, 7]
daily_high = [24, 24, 25, 24, 25, 23, 21, 21, 22, 23, 23, 23]
daily_low = [8, 9, 10, 11, 11, 10, 10, 10, 10, 9, 7, 7]

fig, ax = plt.subplots()
ax.bar(months, precipitation, color='gray')
ax2 = ax.twinx()
ax2.plot(months, daily_high, color='red')
ax2.plot(months, daily_low, color='blue')
plt.show()
```



Explanation: in this figure, we put both precipitation data and temperature data (high and low) together. Since several different data records are in the same place, we use the `subplots()` function to make them appear in one single figure instead of multiple ones. We also choose to add paramaters such as color to differentiate presented data, so that the overall display can convey the information to the readers more effectively.

In summary, first and foremost, any data visualization must accurately and completely reflect the original data. Additionally, scientists should also make a reasonable choice of the data visualization form. In our example of present Addis Ababa's climate data, we use both a line chart and a histogram to represent a series of data. Both the line chart and the histogram are possible for displaying the differences between any two data points. However, a line chart is usually adopted to reflect the changes whereas a histogram may be more effective to tell the absolute value of data.

We have demonstrated the use of Python's visualization library, Watplotlib. We must realize that choosing a suitable visualization method is equally as important as choosing a right visualization tool. Furthermore, annotations such as labels and legends are necessary to help people understand the visualization. For making a good data visualization, there are other factors for us to consider, such as chart size, color selections and uses, design styles, animations, and so on.

3.3.6 Exercises

Exercises

1. An Overview of Data and Data Analysis

1) Show three different sources of data in the real world and outline the main challenge of analyzing each dataset.

2) Imagine that a system receives data at 5 Giga Bytes per second (i.e. 5GB/s) for a scientific project. However, the scientists must concurrently process all the data because the store-now-process-later option is unavailable. Moreover, these scientists do not have any single piece of equipment that processes such a large amount of data at that speed. The only systems they have can process 1 GB of data per second (1GB/s). What is your solution to this data processing problem?

3) Does structured data have higher quality than unstructured data? If so, justify your answer. If not, give a counterexample.

2. Data Operations

1) Write a Python program that can take a limitless number of user input values but let the program output a maximum of first 6 values. You can freely design and determine how the program takes each value and how it stops accepting user inputs. For example, if the user inputs 1, 2, 3, 4, the program outputs 1, 2, 3, 4. If the user inputs 1, 3, 5, 7, 9, 2, 4, 6, 8, 0, the program outputs 1, 3, 5, 7, 9, 2.

2) There is a .csv file that stores data like a spreadsheet. There are 100 lines in total. Each line has 25 commas that are used to separate data. This .csv file also contains a title line (in the first row) and an index column (in the first column). The titles and indices are not part of the actual and meaningful data records. Only the remaining fields (i.e. data below the title line and right to the index column) are the actual data records. How many actual data records can this .csv file store?

3) Assume that an end user inputs a line of data. That line contains all numbers that are separated by comma. For example, the user input from the keyboard is: 150,200,250,300,400,500,50,5,0,1,2 There is a comma (no space) between any two numbers. There is no comma before the first number or after the last number. Write a Python program to read the entire line and parse the data in a way that treats the numbers separately. The program should produce the output with each number in one line, like the following:

```
150
200
250
300
400
500
50
5
0
1
2
```

5) A well-formatted .csv file should have only one kind of data separator, such as commas. What if the original data file contains commas and semicolons both as data separators? Outline a method to correctly process that kind of file.

3. Introduction to Pandas

1) We will be using the World Population Dataset for this question. The dataset was originally obtained from the World Bank of the United Nations: https://api.worldbank.org/v2/en/indicator/SM.POP.TOTL?downloadformat=csv

Some pre-processing steps have already been taken for the purpose of making this question, including the removal of unnecessary rows from the original file. You can use our prepared file world_population.excel for the data analysis purpose.

a) Read the data file.

A dataset is often so large that we do not want to display everything from it. Sometimes, we just want to take a glance at the structure of the dataset by viewing the first few rows.

b) Display the first 5 rows of data.

Particular rows may be selected for displaying purposes. For example, we may be interested in knowing the population values for a specific country or multiple countries.

c) Display the population values for the country Ethiopia.

d) Select and display all countries and regions whose population is over 10 million in the year of 2000.

2) Write one Python program that calculates the following values from a given set of numbers: 10301, 4994, 8872, 9624, 3666, 924, 73712, 3823, 55900, 62, 6498, 852, 24540, 421, 67891, 924, 80192, 3667, 494, 1788. You can choose any data structure for the storage and the processing of these numbers. You will then choose and apply any suitable functions for the calculations.

   a) the number count, i.e. how many elements in this set.

   b) the sum of all these numbers.

   c) the minimum value from this set.

   d) the maximum value from this set.

   e) the median value from this set.

   f) the mode of the elements in this set.

   g) the standard deviation from this set.

   h) the variance (defined as the square of the standard deviation from #7).

4. Basics of matplotlib

If a statistician wants to analyze the data of total births by month in a region, i.e. January: 253, February: 229, March: 247, … What charts would you suggest to use in order to display the monthly records in data visualization? What type of chart is more suitable or less suitable, and for what kind of analysis purpose? Hint: this is a semi-open-ended question. The selection of a chart is determined by a set goal.

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

3.4 Solution to Exercises

3.4.1 Introduction to Python

3.4.2 Working with Data

An Overview of Data and Data Analysis

Exercise 1

Answers may vary. Data of chess moves from human players: sophisticated algorithms are needed to understand the meaning of the moves and the impact of each move towards the winning of the chess game. Real-time data from background radiation: it is nearly impossible to predict the next minute's exact radiation level by "analyzing" the historical values, though an average level over a time period is available. Written texts of a now-extinct human language: because there is no gold-standard human verification, a machine's interpretation of such a language can be less convincing.

Exercise 2

There are several suggested options, but all depend on the nature of the specific data. One can implement parallel systems, consider data compression if possible, or just prioritize the volume with less accurate results, etc.

Exercise 3

Not really. The structure and the quality of data are two different things. One individual may use a numeric value from 0-5 to self-record the tastiness level for a certain type of food that he or she eats every day. However, such kind of record can be very much subjective and does not imply a high quality or accuracy.

Data Operations: Importing and Exporting, Reading and Writing Data

Exercise 1

The purpose of making this question is to provide a general solution in case that one does not want a program to display too many data at a time.

All user entered elements to be stored in a list

list_numbers = list()

Accepting user inputs

user_input = input()

Continue to accept user inputs until user types STOP

while user_input != 'STOP': list_numbers.append(user_input) user_input = input()

Displaying the smaller of: number of user entered elements or 6

records_to_display = min(len(list_numbers), 6)

Using a loop to display uer input values

for i in range(records_to_display): print(list_numbers[i])

Exercise 2

Except the first line, all other 99 lines (rows) have actual data. Usually, there is no comma after last data record in each line. There will be 26 records in each line (row). However, since the first column is storing an index value, there will be 25 actual data records in each line (row). Therefore, the total number of records in this .csv file can store (100-1) * (25+1-1) = 2475.

Exercise 3

We must treat each element as a number type value. The following code does not work, though the output looks okay.

user_input_line = input() output_format = user_input_line.replace(“,” , “\n”) print(output_format)

The correct function that we should use is the split() function.

user_input_line = input()

output_list = user_input_line.split(“,”) output_numbers = []

for i in range(len(output_list)): # adding and displaying the numbers as number type values output_numbers.append(int(output_list[i])) print(output_numbers[i])

Exercise 4

One way to solve the problem is to replace all other symbols with the same separator such as a comma throughout the file.

```
#. The separator symbol (e.g. semicolon) to be replaced
separator_other = ";"
#.The separator symbol (e.g. comma) to replace other separator symbols
separator_comma = ","
#. Assume that all_data is the entire text from the file. Using .replace() function␣
 ↪to update ; with ,
all_data = all_data.replace(separator_other, separator_comma)
```

Another way is to recognize both commas and semicolons as data separators.

```
#. Example: In the sep parameter part, put both , and ; as separators.
df = pd.read_csv("file_name.csv", sep = ",;")
```

Introduction to Pandas

Exercise 1

```
# a) Read the data file
import pandas as pd
population_raw_data = pd.read_excel('world_population.xlsx')

# b) print the first five lines raw data
population_raw_data.head(5)
```

```
                  Country Name Country Code  \
0                        Aruba          ABW
1  Africa Eastern and Southern          AFE
2                  Afghanistan          AFG
3   Africa Western and Central          AFW
4                       Angola          AGO

                      Indicator Name Indicator Code      1960  1961  1962  \
0  International migrant stock, total    SM.POP.TOTL    7893.0   NaN   NaN
1  International migrant stock, total    SM.POP.TOTL       NaN   NaN   NaN
2  International migrant stock, total    SM.POP.TOTL   46468.0   NaN   NaN
3  International migrant stock, total    SM.POP.TOTL       NaN   NaN   NaN
4  International migrant stock, total    SM.POP.TOTL  122089.0   NaN   NaN

   1963  1964     1965  ...  2013  2014        2015  2016  2017  2018  2019  \
0   NaN   NaN   7677.0  ...   NaN   NaN     36114.0   NaN   NaN   NaN   NaN
1   NaN   NaN      NaN  ...   NaN   NaN  10406165.0   NaN   NaN   NaN   NaN
2   NaN   NaN  49535.0  ...   NaN   NaN    382365.0   NaN   NaN   NaN   NaN
3   NaN   NaN      NaN  ...   NaN   NaN   8270665.0   NaN   NaN   NaN   NaN
4   NaN   NaN  77075.0  ...   NaN   NaN    106845.0   NaN   NaN   NaN   NaN

   2020  2021  2022
0   NaN   NaN   NaN
1   NaN   NaN   NaN
2   NaN   NaN   NaN
3   NaN   NaN   NaN
4   NaN   NaN   NaN

[5 rows x 67 columns]
```

```
# c) Display the population values for Ethiopia

Ethiopia=population_raw_data[population_raw_data["Country Name"] == "Ethiopia"]
Ethiopia
```

```
   Country Name Country Code                      Indicator Name  \
72     Ethiopia          ETH  International migrant stock, total

   Indicator Code      1960  1961  1962  1963  1964      1965  ...  2013  \
72    SM.POP.TOTL  393260.0   NaN   NaN   NaN   NaN  383551.0  ...   NaN

    2014       2015  2016  2017  2018  2019  2020  2021  2022
72   NaN  1072949.0   NaN   NaN   NaN   NaN   NaN   NaN   NaN

[1 rows x 67 columns]
```

```
population_raw_data.columns
```

```
Index([  'Country Name',   'Country Code', 'Indicator Name', 'Indicator Code',
                   1960,             1961,             1962,             1963,
                   1964,             1965,             1966,             1967,
                   1968,             1969,             1970,             1971,
                   1972,             1973,             1974,             1975,
                   1976,             1977,             1978,             1979,
                   1980,             1981,             1982,             1983,
                   1984,             1985,             1986,             1987,
                   1988,             1989,             1990,             1991,
                   1992,             1993,             1994,             1995,
                   1996,             1997,             1998,             1999,
                   2000,             2001,             2002,             2003,
                   2004,             2005,             2006,             2007,
                   2008,             2009,             2010,             2011,
                   2012,             2013,             2014,             2015,
                   2016,             2017,             2018,             2019,
                   2020,             2021,             2022],
      dtype='object')
```

```
# d) Display all countries/regions whose population was over 10 million in the year␣
↪2000
df = population_raw_data[["Country Name",2000]]
large=df[df[2000]>10000000]
large
```

	Country Name	2000
7	Arab World	16216741.0
62	Early-demographic dividend	38155943.0
63	East Asia & Pacific	15674381.0
64	Europe & Central Asia (excluding high income)	27780057.0
65	Europe & Central Asia	64005283.0
68	Euro area	26129209.0
73	European Union	29602637.0
74	Fragile and conflict affected situations	10005606.0
95	High income	100949072.0
98	Heavily indebted poor countries (HIPC)	10632626.0
102	IBRD only	51103969.0
103	IDA & IBRD total	72738421.0
104	IDA total	21634452.0
107	IDA only	13832671.0
135	Least developed countries: UN classification	10073307.0
139	Lower middle income	31018425.0
140	Low & middle income	70316148.0
142	Late-demographic dividend	30615531.0
153	Middle East & North Africa	20000241.0
156	Middle income	63267150.0
170	North America	40343635.0
181	OECD members	86944671.0
191	Pre-demographic dividend	10751023.0
198	Post-demographic dividend	92403421.0
202	Russian Federation	11900297.0
204	South Asia	12474215.0
215	Sub-Saharan Africa (excluding high income)	13462901.0
217	Sub-Saharan Africa	13469475.0

(continues on next page)

(continued from previous page)

```
231   Europe & Central Asia (IDA & IBRD countries)   29190606.0
240                         South Asia (IDA & IBRD)   12474215.0
241      Sub-Saharan Africa (IDA & IBRD countries)   13469475.0
249                            Upper middle income   32248725.0
251                                  United States   34814053.0
259                                          World  172278883.0
```

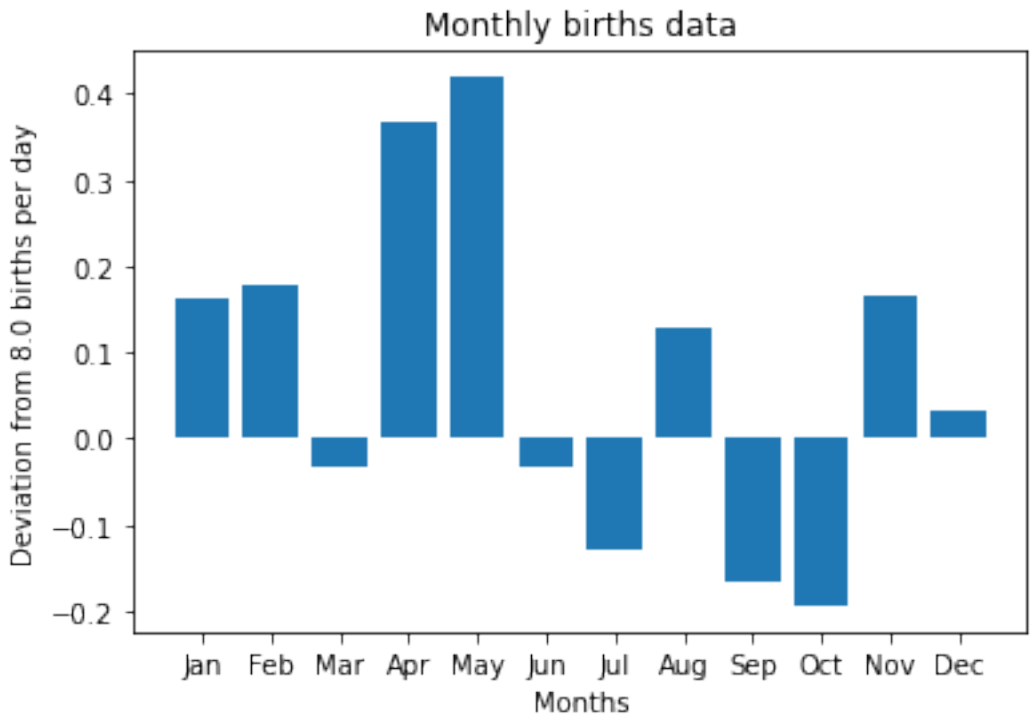
Exercise 2

```
20
20
359145
62
80192
4408.5
0    924
dtype: int64
27283.438618124135
744386022.8289474
```

Basics of MATPLOTLIB

A bar chart may be considered; however, some modifications from the original bar chart are necessary. The values of total births in different months are likely close to each other. A simple bar chart that is showing the absolute values may not present the difference of values very well. Different months have different number of days. It is likely that February has the smallest number, but such a fact does not mean "February is the month that people avoid giving births."

```
One suggestion is that we use the number of births per day in a month to represent␣
↪data.

The following chart is produced with the reflection of modifications.
```



3.5 JNB Lab: Introduction to Python

This lab accompanies the chapter "Introduction to Python." The goal of this lab is to discover a sampling of Python features and programming concepts by way of example. The concepts covered in this lab are

- Recursive algorithms
- Python list comprehensions
- Optimization by way of mathematical transformation
- Programming as data plus operations
- Classes

The running example on which we hang these concepts is modeling a polynomial function, that is, a function in the form of

$$f(x) = c_0 + c_1 x + c_2 x^2 + \cdots c_n x^n = \sum_{i=0}^{n} c_i x^i$$

for example, $f(x) = 3 + 4x - 7x^2 + 5x^3$.

In the chapter itself, we suggested that a function like this can be represented by a list of coefficients—in this case, `[3.0, 4.0, -7.0, 5.0]`. Think for a moment about this idea of *representation*: We have taken the essential information from the abstract concept of a polynomial function and reduced it to a form such that that information can be stored in a Python type. The number of terms—the *degree* of the polynomial—can be determined by the length of the list, and each coefficient's position in the list indicates the power of x that the coefficient corresponds to. If there is any power of x for which this polynomial doesn't have a term, we can represent that with a `0.0` in the corresponding position in the list. But it this list doesn't act like a function—you cannot apply it to a specific value of x. Instead, we need to have separate Python functions to compute anything about this polynomial function.

In the section we saw the following function for evaluating a polynomial represented as a list of coefficients

```
def evaluate_polynomial(coefficients, x) :
    result = 0
    for i in range(len(coefficients)) :
        result += coefficients[i] * x**i
    return result
```

This lab steps you through variations on this function.

3.5.1 Doing it recursively

Recursion is defining something in terms of a smaller version of itself. Many things in both math and computation afford simple self-referential definitions. The trick is to identify a part that is similar to a whole. Consider polynomials—notice that in our example, the terms $3+4x-7x^2$ by themselves make a polynomial. Suppose we define $g(x) = 3+4x-7x^2$, then our original function f can be rewritten as

$$f(x) = g(x) + 5x^3$$

…that is, a (smaller) polynomial plus the term of degree 3. In general, a polynomial of degree $n \geq 0$ is

$$f(x) = \begin{cases} c_0 & \text{for some constant } c_0 & \text{if } n = 0 \\ \hat{f}(x) + c_n x^n & \text{for some polynomial } \hat{f} \text{ of degree } n-1 \text{ and constant } c_n & \text{otherwise} \end{cases}$$

For a given list, such as one storing the coefficients of a polynomial, we can grab *all but the last* element using slicing and negative indexing:

```
coeffs = [3.0, 4.0, -7.0, 5.0]
print(coeffs)
print(coeffs[:-1])
```

(Notice that the *degree* of the polynomial and the *length* of the list representing the polynomial are not equal. Rather, the length of the list, or the number of coefficients, is one greater than the degree. A zero-degree polynomial has one coefficient, namely the constant term; a list representing a zero-degree polynomial has length one, since it contains only that constant. Moreover, the degree of the polynomial is a valid index into the list of coefficients, since a polynomial of degree n has an nth coefficient.)

Thus we have a conceptual way to evaluate a polynomial in terms of evaluating a smaller polynomial, and we know how to extract a smaller polynomial from one given as a list. We have the makings of a recursive process for polynomial evaluation. What we still need is a sense of how to make a recursive function in code. Let's pause this problem and try a simpler one: Suppose we want to compute the result of multiplying all the elements in a list together, which we might call the *product* of a list. Python has a built-in function `sum` which takes a list and computes the sum of all the elements in that list, but it doesn't have a similar built-in `product` function for lists. We'll need to write it ourselves. But notice that the product of all the elements in a list is equal to the product of all but the last element followed by multiplying that last element with the product of the rest. What we need to check for is whether the list is empty, in which case the list elements have a default product of 1.

```
def product(xx) :
    if len(xx) == 0:
        return 1
    else :
        return product(xx[:-1]) * xx[-1]
```

```
product([2, 4, 3, 9])
```

We call the special case where we can find the answer directly the *base case*, and the case where the function calls itself the *recursive case*. To follow how this works, inspect this revision that includes tracing code so that we can follow how the function executes.

```
def product(xx, original_length) :
    indent = '  ' * (original_length-len(xx))
    print(indent + "product called with " + str(xx))
    if len(xx) == 0:
        result = 1
    else :
        result =  product(xx[:-1], original_length) * xx[-1]
    print(indent + "returning with result " + str(result))
    return result
```

```
product([2, 4, 3, 9], 4)
```

However, this "traced" version is just to build your intuition (and faith) about how recursion works. This is not the mental model you should have when thinking about recursive algorithms. Instead, your attention should be on $f(x) = \hat{f}(x) + c_n x^n$, that is, how to relate a (bigger) problem in terms of a (smaller) problem.

Exercises

1.1

Write a new version of the Python function `evaluate_polynomial` that does not use a while loop but instead uses recursion. A stub for this function is given below. You may assume that the minimum size of the list is 1, which is checked by an assertion in the stub:

```
def evaluate_polynomial(coefficients, x) :
    assert len(coefficients) > 0
    # Add code here
```

1.2

In the function `product`, why is the product of an empty list 1 instead of 0?

3.5.2 Doing it by comprehension

Although recursion is a useful tool of abstraction that allows us to simplify many algorithmic tasks, neither the recursive version nor the original iterative (i.e., "using a loop") version of `evaluate_polynomial` follows the native Python idiom for identifying patterns that can be expressed compactly, like what we have in the mathematical formula for a polynomial. For that, we need to use a Python construct called a *list comprehension*, which allows us to make a new list using a pattern.

Python comprehensions are modeled after mathematical notation used to specify a sequence. Consider the sequence of powers of two:

$$1, 2, 4, 8, 16, 32, ...$$

In this we're considering the first item in the sequence to be $2^0 = 1$, which plays well with zero-based indexing. Now, instead of listing the elements in the sequence, we can specify this sequence using a pattern:

$$2^i \text{ for } i \in \mathbb{W}$$

where $\mathbb{W}$ stands for *whole numbers*, that is, non-negative integers. This is an infinite sequence, but we can specify a finite portion of it by setting an upper bound for i. We let $\mathbb{Z}_n$ stand for nonnegative integers less than n, for example $\mathbb{Z}_5 = \{0, 1, 2, 3, 4\}$. Now we can say

$$2^i \text{ for } i \in \mathbb{Z}_8 = 1, 2, 4, 8, 16, 32, 64, 128$$

We express the same idea in Python:

```
[2**i for i in range(8)]
```

So far we have seen how to produce the items in a sequence using a comprehension. It is only one step further to turn this into a summation: Use the built-in Python function `sum` alluded to above.

```
sum([2**i for i in range(8)])
```

Turning back to our running example, we notice that the terms in a polynomial are just the elements in a sequence specified by the pattern $c_i x^i$ for $i \in \mathbb{Z}_{n+1}$ where n is the degree of the polynomial.

Exercises

2.1

Write new version of the Python function `evaluate_polynomial` that uses neither loop nor recursion but instead uses `sum` and a list comprehension. A stub for this function is given below. As before, you may assume that the minimum size of the list is 1.

```
def evaluate_polynomial(coefficients, x) :
    assert len(coefficients) > 0
    # Add code here
```

2.2

When we specified the terms of a polynomial by a pattern, why did we say "for $i \in \mathbb{Z}_{n+1}$" instead of "for $i \in \mathbb{Z}_n$"?

3.5.3 Doing it efficiently: Horner's rule

In the section "Thinking through Algorithms", we considered not only what an algorithm does, but how efficiently it does it, which we measured by counting how many steps the algorithm took in terms of the size of the input. We can measure the computational cost of these algorithms for evaluating a polynomial by counting the number of multiplications, which we'll use as a proxy for the entire cost.

For the ith term, $c_i x^i$, we perform i multiplications. For example, $3x^4 = 3 \cdot x \cdot x \cdot x \cdot x$. This means that in terms of the degree of the polynomial n, the number of multiplications is an arithmetic sum, $0 + 1 + 2 + \cdots n$, which is *quadratic* in n:

$$\sum_{i=0}^{n} = 0 + 1 + 2 + \cdots + n = \frac{(n+1)n}{2} = \frac{1}{2}(n^2 + n)$$

All of our solutions so far have the same cost by this measure, because they all perform the same multiplications. (This is assuming that the Python `**` operator simply does a sequence of multiplications—which is not true but is still a reasonable model for our present purposes.) But, one might suppose, aren't those multiplications inherent to the problem? Surely we can't reduce that cost?

In fact, we can write an algorithm for evaluating polynomials that performs fewer multiplications, and the first insight into how is to notice that our algorithms so far have all done redundant work when computing the powers of x. For example, if one of our algorithms needs to compute x^4, it does it from scratch. However, on a previous step they had already computed x^3, prior work that the algorithm fails to capitalize on.

We can rewrite our original, iterative version so that it saves its exponentiation work from the previous iteration and eliminate this unnecessary work:

```
def evaluate_polynomial(coefficients, x) :
    result = 0
    x_pow = 1 # This is always x^i; it's the power of x for the next iterations
    for i in range(len(coefficients)) :
        result += coefficients[i] * x_pow
        x_pow *= x
    return result
```

But there is also a way we can manipulate the formula algebraically to achieve this. We repeatedly pull out a factor of x from portions of the polynomial:

$$\begin{aligned} c_0 + c_1x + c_2x^2 + c_3x^3 + \cdots c_{n-1}x^{n-1} + c_nx^n &= c_0 + x(c_1 + c_2x + c_3x^2 + \cdots c_{n-1}x^{n-2} + c_nx^{n-1}) \\ &= c_0 + x(c_1 + x(c_2 + c_3x + \cdots c_{n-1}x^{n-3} + c_nx^{n-2})) \\ &= c_0 + x(c_1 + x(c_2 + x(c_3 + \cdots c_{n-1}x^{n-4} + c_nx^{n-3})) \\ &\vdots \\ &= c_0 + x(c_1 + x(c_2 + x(c_3 + x(\cdots x(c_{n-1} + x(c_n))\cdots)))) \end{aligned}$$

This formula is known as **Horner's Rule**.

Exercises

3.1

Finish the following Python function to implement `evaluate_polynomial` using Horner's Rule *iteratively*. Notice that the loop in the given code counts *down* from n using variable `i`. For each step in the process, we execute a piece of the formula in the form of

$$c_i + x(\text{result so far})$$

```
def evaluate_polynomial(coefficients, x) :
    result = 0
    for i in reversed(range(len(coefficients))) :
        # add code here
    return result
```

3.2

Write another version of `evaluate_polynomial` that uses Horner's rule *recursively*. To catch the recursive structure of this problem, look again at the first step that we took when factoring:

$$c_0 + c_1x + c_2x^2 + c_3x^3 + \cdots c_{n-1}x^{n-1} + c_nx^n = c_0 + x(c_1 + c_2x + c_3x^2 + \cdots c_{n-1}x^{n-2} + c_nx^{n-1})$$

Notice that $c_1 + c_2x + c_3x^2 + \cdots c_{n-1}x^{n-2} + c_nx^{n-1}$ is itself a polynomial. It has degree $n - 1$. We can retrieve that (sub-)polynomial from a list that represents the larger polynomial, grab the slice beginning at index 1. As before, we provide a stub:

```
def evaluate_polynomial(coefficients, x) :
    assert len(coefficients) > 0
    # add code here
```

3.3

How many multiplications does Horner's rule perform when given a polynomial with degree n?

3.5.4 Making other operations (requires calculus)

Programming is about representing information and making operations to, well, operate on that information. We have been using a list of coefficients to represent information about polynomials, and we have one operation that we have implemented several ways. However, *evaluating* mathematical functions is not the only thing we ever want to do with them. From calculus, we know that we can also differentiate or integrate a function. Remember that for a polynomial in the form we have been using, its derivative is

$$f'(x) = c_1 + 2 \cdot c_2 x + 3 \cdot c_3 x^2 + \cdots + n \cdot c_n x^{n-1}$$

The indefinite integral (or antiderivative) is (with arbitrary constant c^*)

$$F(x) = c^* + c_0 x + \frac{1}{2} c_1 x^2 + \frac{1}{3} c_2 x^3 + \frac{1}{4} c_3 x^4 + \cdots + \frac{1}{n+1} c_n x^{n+1}$$

The definite integral from a to b is

$$\int_a^b f(x)\,dx = F(b) - F(a)$$

Using this refresher, we proceed directly to the…

Exercises

4.1

Finish the following Python function to compute the derivative of a given polynomial, represented as a list of coefficients. Notice that this function should return another polynomial, represented as a list. Use a list comprehension to do this.

```
def differentiate_polynomial(coefficients) :
    # add code here
```

4.2

Finish the following Python function to compute the indefinite integral of a given polynomial, represented as a list of coefficients. Like the previous problem, this function should return another polynomial. Use a list comprehension.

```
def integrate_polynomial_indef(coefficients) :
    # add code here
```

4.3

Finish the following Python function to compute the definite integral of a given polynomial with lower and upper bounds. The function you write for this exercise should call your function from the previous exercise.

```
def integrate_polynomial_def(coefficients, a, b) :
    # add code here
```

3.5.5 Making a class

Programming is about representing information and making operations on that information. What makes a *class* different from other types or programming modules is that it allows us to package information and the operations on the information into one encapsulated entity. Consider the following class for representing polynomials which wraps a list of coefficients together with the operation to evaluate that polynomial (we used the iterative Horner's rule version). We don't put that functionality into a method called `evaluate_poly` but instead into a special method called `__call__`. This name gets special recognition in Python, making the class *callable*. That means that instances of this class can be used as functions, which makes sense in this case because polynomials are mathematical functions.

```
class Polynomial :
    def __init__(self, coeffs) :
        self.coefficients = coeffs.copy()

    def __call__(self, x) :
        result = 0
        x_pow = 1 # This is always x^i; it's the power of x for the next iterations
        for i in range(len(self.coefficients)) :
            result += self.coefficients[i] * x_pow
            x_pow *= x
        return result

    def degree(self) :
        return len(self.coefficients)-1

    # Another special function, this one makes a nice string representation
    # of the object
    def __str__(self) :
        if len(self.coefficients) == 0 :
            return '0.0'
        else :
            to_return = str(self.coefficients[0])
            for i in range(1, len(self.coefficients)) :
                to_return += ' + ' + str(self.coefficients[i]) + 'x'
                if i > 1 :
                    to_return += '^' + str(i)
            return to_return


    # The next three are for the exercises
    def differentiate(self) :
        pass

    def integrate_indef(self) :
        pass

    def integrate_def(self, a, b) :
        pass
```

Let's demonstrate how a callable can be used.

```
f = Polynomial([3.0, 4.0, -7.0, 5.0])
print(f)
```

```
f(2.5)
```

Exercises

5.1

Finish the class `Polynomial`. You can reuse code from previous exercises, adjusting them to make sense in the context of the class. The methods `differentiate` and `integrate_indef` should return `Polynomials`.

3.6 JNB Lab Solutions

Exercise 1.1

```
def evaluate_polynomial(coefficients, x) :
    assert len(coefficients) > 0
    if len(coefficients) == 1 :
        return coefficients[0]
    else :
        return evaluate_polynomial(coefficients[:-1],x) + coefficients[-1] *␣
↪x**len(coefficients)
```

Exercise 1.2

Treating the product of an empty list as 1 makes the recursion work. If the base case returned 0, then all other products would be 0.

Exercise 2.1

```
def evaluate_polynomial(coefficients, x) :
    assert len(coefficients) > 0
    return sum([coefficients[i] * x**i for i in range(len(coefficients))])
```

Exercise 2.2

$\mathbb{Z}_{n+1}$ includes n itself, which $\mathbb{Z}_n$ doesn't. Moreover, n is a valid index to the list of coefficients. Mathematically, $\mathbb{Z}_{n+1}$ is equivalent to `range(len(coefficients))` in code, since the length of the list (which is also the number of coefficients) is one more than the degree of the polynomial.

Exercise 3.1

```
def evaluate_polynomial(coefficients, x) :
    result = 0
    for i in reversed(range(len(coefficients))) :
        result = coefficients[i] + x * result
    return result
```

Exercise 3.2

```
def evaluate_polynomial(coefficients, x) :
    assert len(coefficients) > 0
    if len(coefficients == 1) :
        return coefficients[0]
    else :
        return coefficients[0] + x * evaluate_polynomial(coefficients[1:], x)
```

Exercise 3.3

$n - 1$.

Exercise 4.1

```
def differentiate_polynomial(coefficients) :
    return [coefficients[i]/i for i in range(1,len(coefficients))]
```

Exercise 4.2

```
def integrate_polynomial_indef(coefficients) :
    # 17.0 is an arbitrary constant. Happy Cicada Brood XIII Year, 2024
    return [17.0] + [coefficients[i]/(i+1) for i in range(len(coefficients))]
```

Exercise 4.3

```
def integrate_polynomial_def(coefficients, a, b) :
    antideriv = integrate_polynomial_indef(coefficients)
    return evaluate_polynomial(antideriv, b) - evaluate_polynomial(antideriv, a)
```

Exercise 5.1

```
class Polynomial :
    def __init__(self, coeffs) :
        self.coefficients = coeffs.copy()

    def __call__(self, x) :
        result = 0
        x_pow = 1 # This is always x^i; it's the power of x for the next iterations
        for i in range(len(self.coefficients)) :
            result += self.coefficients[i] * x_pow
            x_pow *= x
        return result

    def degree(self) :
        return len(self.coefficients)-1

    # Another special function, this one makes a nice string representation
    # of the object
    def __str__(self) :
        if len(self.coefficients) == 0 :
            return '0.0'
        else :
            to_return = str(self.coefficients[0])
            for i in range(1, len(self.coefficients)) :
                to_return += ' + ' + str(self.coefficients[i]) + 'x'
                if i > 1 :
                    to_return += '^' + str(i)
            return to_return


    # The next three are for the exercises
    def differentiate(self) :
        return Polynomial([self.coefficients[i]/i for i in range(1,len(self.
↪coefficients))])

    def integrate_indef(self) :
        return Polynomial([17.0] + [self.coefficients[i]/(i+1) for i in␣
↪range(len(selfcoefficients))])

    def integrate_def(self, a, b) :
        antideriv = self.integrate_indef()
        return antideriv(b) - antiderive(a)
```

CHAPTER 4

EXPLORATORY DATA ANALYSIS - Jonathan Zhu

Note

The material on exploratory data analysis (EDA) was developed by Jonathan Zhu (graduate student, Carnegie Mellon University). Paul Isihara and Samuel Carlson assisted with the JNB lab.

4.1 Introduction to Part I

Exploratory Data Analysis (EDA) is the process of preliminarily working through data to correct or remove any errors and find preliminary insights about the dataset. We will cover some of the basic procedures behind EDA in this portion of the chapter. Specifically, we will use a sample dataset to cover the following topics:

- Basic data examination
- Removing N/A values
- Plotting with `matplotlib`
- Basic statistical tests

This section will start with some basic information on EDA and what is done in it; it requires knowledge of the previous sections, especially the Python programming guide. We will focus primarily on the practical methods for performing certain aspects and less so the mathematical theory behind it; later sections will cover the theoretical foundation behind any statistics performed in this section. Additionally, the second half of the chapter will have more advanced EDA procedures on a different, much larger dataset that relates more closely to real-world problems.

4.1.1 Why is EDA Important?

Often times in the real world, regardless of application, datasets that have to work with will not be perfectly suitable for advanced usage such as machine learning. Many datasets have many errors, missing values, and outliers that will confound your analysis; any analysis done with these errors included will end up biasing your conclusions.

Additionally, it is a bad idea (most of the time) to do advanced procedures with data without knowing anything about the data and how it looks. Data analysis often builds on existing knowledge about data and, in turn, provides new insights about a dataset.

4.1.2 How Do We Do EDA?

The process of EDA varies between datasets as well as between analysis purposes, and EDA does not follow a specific procedure. However, it typically follows the following process:

1. Visually inspect the data
2. Cleaning the data
3. Extracting insights from data

Additionally, EDA does not have to be a one-time process. With more looking at the data and getting insights from it, you will know more about the data and can find even more insights.

4.1.3 About the Dataset

The dataset used for this portion of the chapter can be found at https://archive.ics.uci.edu/dataset/20/census+income. It comes from the UC Irvine Machine Learning Repository and concerns census data from 1994. It was originally intended for machine learning projects to predict whether a given person (represented as one row in the dataset) makes over $50,000 a year.

Knowing the context of data is important in many cases, as the context of the data will inform some of the preliminary analyses you perform on it. We will start looking at the data in the next section.

4.2 Data Investigation I

Often times when we explore data for the first time, we have to spend time looking at what is missing in the data, or what needs to be changed. Data files rarely if ever are perfect for use immediately once you get them. Additionally, it is often unwise to utilize data for predictive modeling or statistical analysis without knowing some of the context for the data and looking at its contents.

Python has a helpful package called `pandas` which is purpose-built for looking at and manipulating data, and a lot of what we will work on in this section will involve using it.

```
import pandas as pd
import numpy as np

#read in data
data = pd.read_csv('census_income.csv')
data.head()
```

```
   age          workclass   fnlwgt   education   education-num  \
0   39          State-gov    77516   Bachelors              13
1   50   Self-emp-not-inc    83311   Bachelors              13
2   38            Private   215646     HS-grad               9
3   53            Private   234721        11th               7
4   28            Private   338409   Bachelors              13

        marital-status          occupation    relationship    race      sex  \
0        Never-married        Adm-clerical   Not-in-family   White     Male
1   Married-civ-spouse     Exec-managerial         Husband   White     Male
2             Divorced   Handlers-cleaners   Not-in-family   White     Male
3   Married-civ-spouse   Handlers-cleaners         Husband   Black     Male
4   Married-civ-spouse      Prof-specialty            Wife   Black   Female

    capital-gain   capital-loss   hours-per-week  native-country  income
0           2174              0               40   United-States   <=50K
1              0              0               13   United-States   <=50K
2              0              0               40   United-States   <=50K
3              0              0               40   United-States   <=50K
4              0              0               40            Cuba   <=50K
```

`data.head()` is a good way to get a good look at the first few rows of the data.

Exercise 2.1

Thinking of the opposite of the head, how do we get the last few rows of the data?

First things first is to get a look at the data. How many rows (samples) are there? How many columns (variables) are there? These numbers correspond to the *shape* of the dataframe.

```
data.shape
```

```
(32561, 15)
```

We can see that this dataset has 32,561 instances of 15 variables. This may seem like a lot, but it is actually relatively small compared to many other modern datasets, like the one used in the advanced EDA portion of this chapter, which has millions of instances!

Secondly, we should get a look at the features themselves. What do they mean? What datatype are they? While we can do the first one by looking at the names of the columns by calling `data.head()`, this does not allow us to do the second. So we can instead do the following:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   age              32561 non-null  int64
 1    workclass       30725 non-null  object
 2    fnlwgt          32561 non-null  int64
 3    education       32561 non-null  object
 4    education-num   32561 non-null  int64
 5    marital-status  32561 non-null  object
```

(continues on next page)

(continued from previous page)

```
 6    occupation      30718 non-null  object
 7    relationship    32561 non-null  object
 8    race            32561 non-null  object
 9    sex             32561 non-null  object
 10   capital-gain    32561 non-null  int64
 11   capital-loss    32561 non-null  int64
 12   hours-per-week  32561 non-null  int64
 13   native-country  31978 non-null  object
 14   income          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

This tells us the datatypes of each variable and gives us a full list of the variables. The `object` datatype corresponds to strings, such as "United States" or "Female", while the `int64` datatype corresponds to integer numbers.

Exercise 2.2

Hypothesize as to what each variable name indicates. Then consult the website where the dataset is found, https://archive.ics.uci.edu/dataset/20/census+income. Do your hypotheses match the actual meaning?

While you may or may not have been able to guess what each variable meant just by looking at the name, it is often the case in the real world that you may have no idea what some variable names mean. This implies that you will have to look to outside resources, especially your data source, to understand what these might mean.

Next, we can get quick summaries of the data. This is very easy to do with `pandas` dataframes.

```
data.describe()
```

```
                age        fnlwgt   education-num   capital-gain  \
count  32561.000000  3.256100e+04    32561.000000   32561.000000
mean      38.581647  1.897784e+05       10.080679    1077.648844
std       13.640433  1.055500e+05        2.572720    7385.292085
min       17.000000  1.228500e+04        1.000000       0.000000
25%       28.000000  1.178270e+05        9.000000       0.000000
50%       37.000000  1.783560e+05       10.000000       0.000000
75%       48.000000  2.370510e+05       12.000000       0.000000
max       90.000000  1.484705e+06       16.000000   99999.000000

        capital-loss   hours-per-week
count   32561.000000     32561.000000
mean       87.303830        40.437456
std       402.960219        12.347429
min         0.000000         1.000000
25%         0.000000        40.000000
50%         0.000000        40.000000
75%         0.000000        45.000000
max      4356.000000        99.000000
```

This will describe all the numerical variables, so long as no N/A values exist in these variables. Note that one N/A value will make all summary statistics in the description table show up as N/A, except for count.

Therefore, you can see that removing N/A values is important. We were lucky to stumble across a dataset with no N/A values in the numerical dataset. However, this dataset does have N/A values in it! This can be problematic.

4.2.1 Cleaning Data

Data cleaning has a lot of meanings in it, but most commonly, the type of cleaning you do to make a dataset more usable for analysis or machine learning is to remove or somehow manage N/A values. These values are empty and can bias any conclusions you make from the data if they are still present when you perform further analysis.

To find the number of N/As in the dataset, we can call a function that only isolates N/A values from the dataset and sums them for every column.

Warning

If you use the raw dataset from the UC Irvine Machine Learning Repository, the N/A values are not actually given as N/A in the raw data, but rather as a question mark `?` and later functions may not count these as N/A values. You can easily replace these with the line `data = data.replace('?', np.nan)`.

```
data.isna().sum()
```

```
age                  0
 workclass        1836
 fnlwgt              0
 education           0
 education-num       0
 marital-status      0
 occupation       1843
 relationship        0
 race                0
 sex                 0
 capital-gain        0
 capital-loss        0
 hours-per-week      0
 native-country    583
 income              0
dtype: int64
```

This dataset is relatively clean and so it has many columns with no missing values; the columns with missing values have relatively few of them. In the case where the N/A values are relatively few, we can remove the rows with an N/A value in any column. Let's try that out and see how it changes our data.

```
#remove rows with missing values
data = data.dropna()
data.shape
```

```
(30162, 15)
```

This simple act dropped around 2000 rows from our dataset. Now we should be ready to use this dataset for further analysis and get insights from it, which we discuss in the next section.

4.2.2 More Exercises

Exercise 2.3

Re-calculate the summary statistics after dropping N/A values and describe any changes.

Exercise 2.4

To describe categorical data, we can have the program print out the different categories within the variable, and the number of instances of each category. Try this out for the `education` column in the dataset.

Exercise 2.5

(Bonus) Write a function that can do Exercise 2.4 for any column of your choosing.

4.3 Plotting and Statistical Testing I

In this section, we will go over some basic plots we can perform with `matplotlib`. This will involve quite a bit of review of content from the section on using `matplotlib` in the Python Programming Guide.

At a basic level, there are a few plots that are widely used in EDAs and for analyses in general. These plots depend on the data that you want to include in the plot, and we will go over some of the main types here.

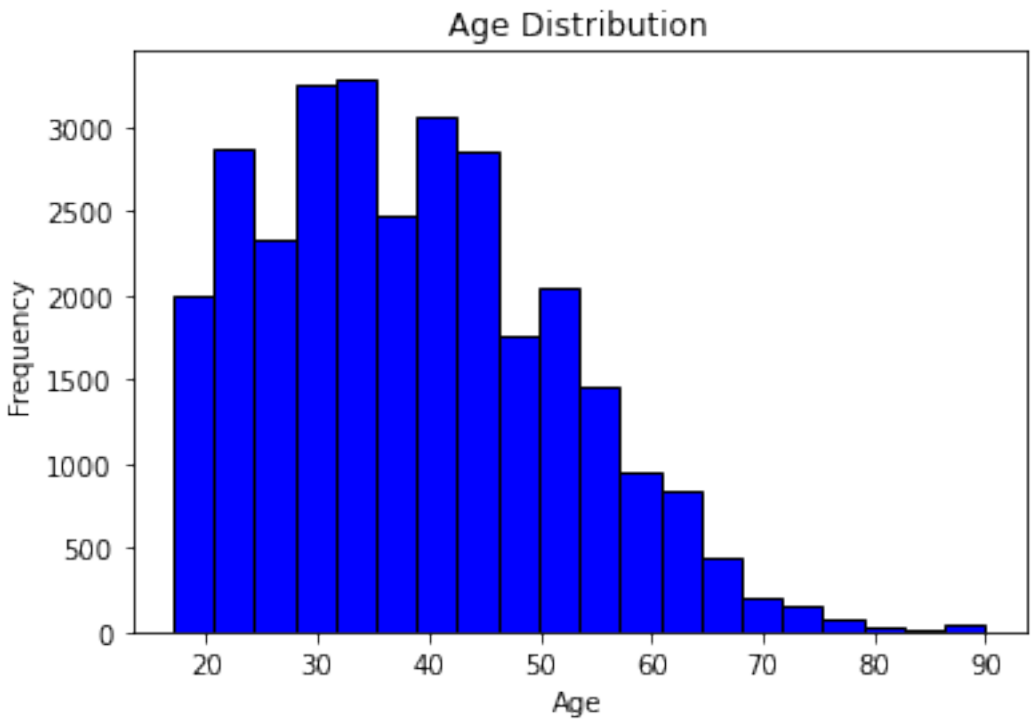
4.3.1 Plotting

One Numerical Variable: Histograms

When visualizing one single numerical variable, you will want to use a histogram. Histograms visualize the frequency of each value across a numerical variable and give you a sense of how the instances are distributed. This in turn can give insight about how likely certain values might be; to find out more about probability distributions, look to the Probability and Statistical Inference sections.

For now, let's make a histogram of the `age` column.

```
#make a histogram of the age column
plt.hist(data['age'], bins=20, color='blue', edgecolor='black')
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



In making our plot, note that we added labels to each axis and a title to the plot. This is important as our graphs should be able to stand alone and still be somewhat understandable.

We can see that the distribution of ages is skewed right (values tend to be lower) but with a few outliers corresponding to the very elderly.

Exercise 3.1

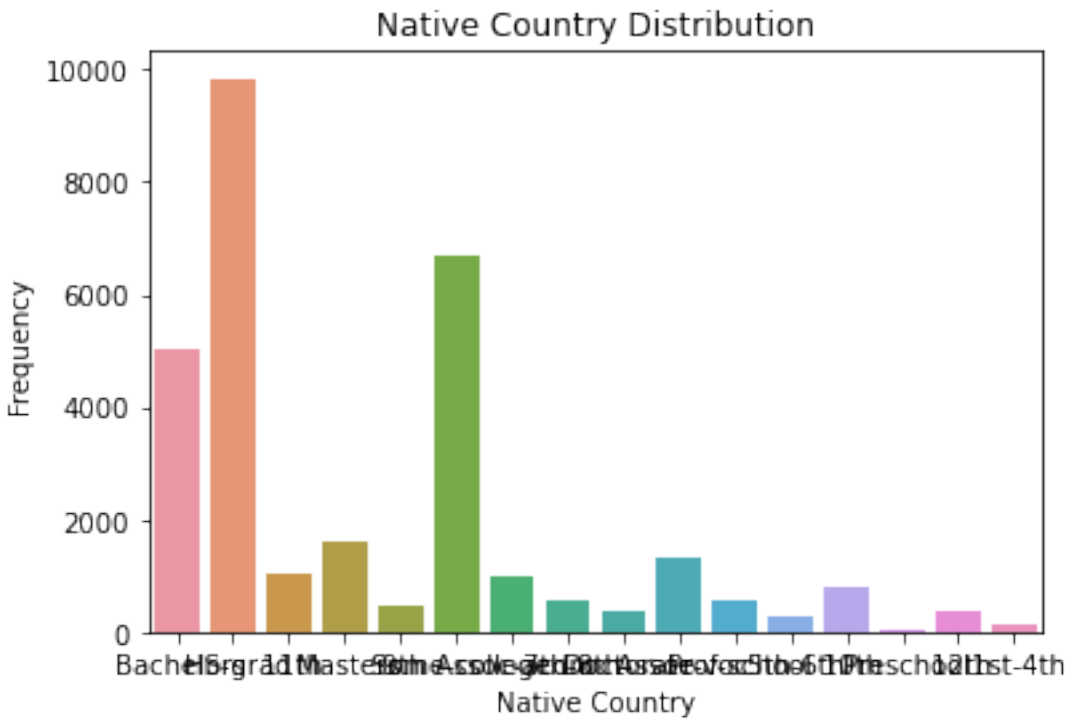
Make another histogram corresponding to a different variable.

One Categorical Variable: Count Plots

To look at the distribution of one categorical variable, we use count plots. These will correspond to the number of instances of each category in a variable, with one bar for each category. While this is possible with `matplotlib`, the `seaborn` package makes for easier and nicer plots much of the time, and this is a case where it is very helpful. It is often used in conjunction with `matplotlib` as well, and we will see that below.

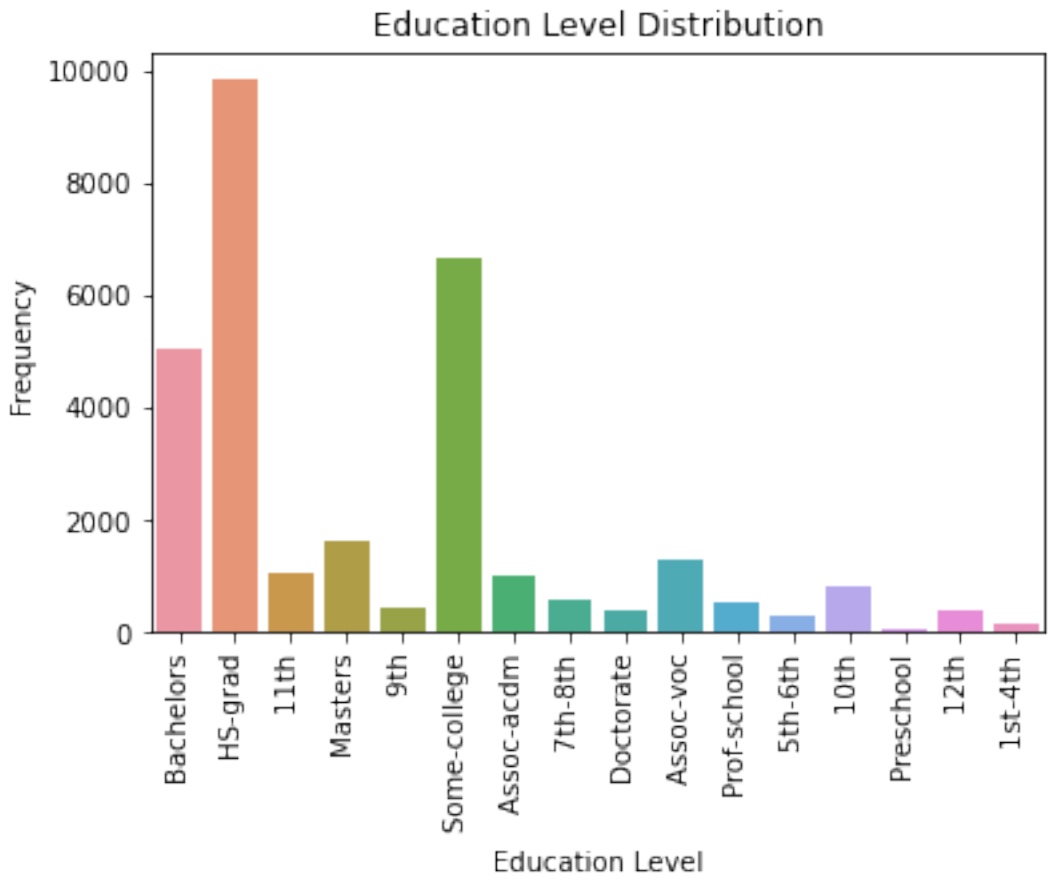
Here, let's make a bar plot of the `education` column.

```
#make a bar plot of the education column
import seaborn as sns

sns.countplot(x = data['education'])
plt.title('Education Level Distribution')
plt.xlabel('Education Level')
plt.ylabel('Frequency')
plt.show()
```



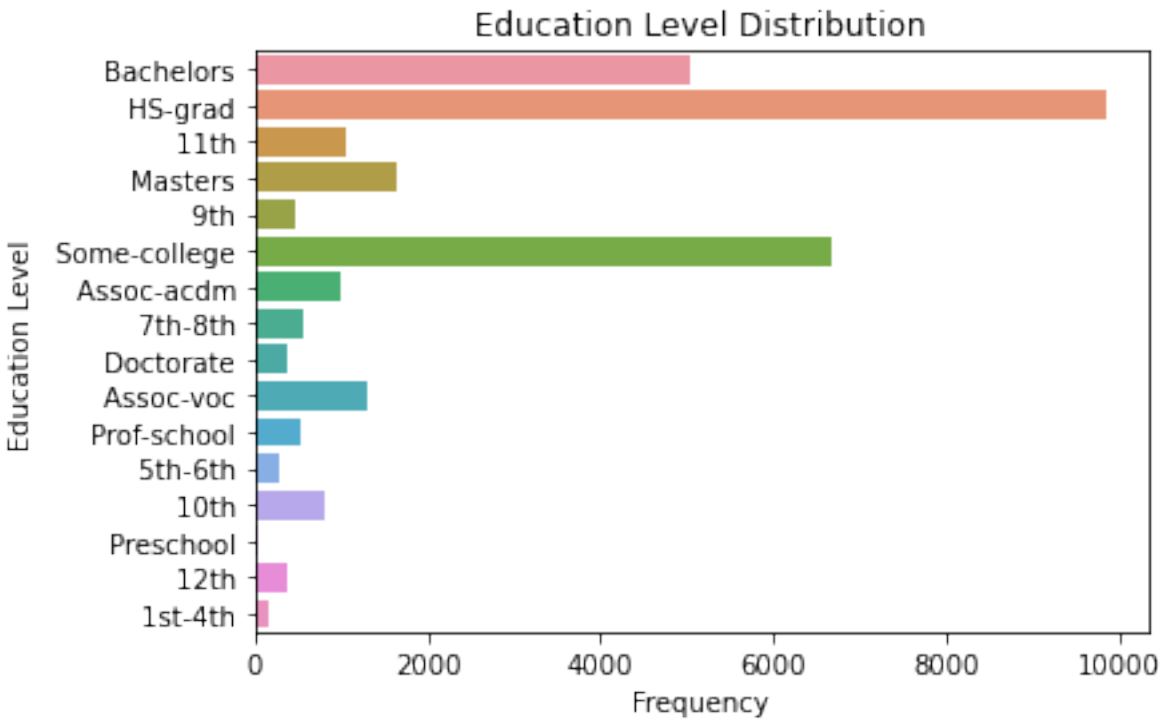
An unfortunate reality of countplots is that they will collide labels with each other for variables with a lot of categories. Fortunately, this is where the collaboration between `seaborn` and `matplotlib` can shine, because the latter allows for rotating the ticks on the axes.

```
sns.countplot(x = data['education'])
plt.title('Education Level Distribution')
plt.xticks(rotation=90)
plt.xlabel('Education Level')
plt.ylabel('Frequency')
plt.show()
```



Much more readable now, but it's easier to read these labels when they are in their normal positions. How about we rotate this entire graph instead?

```
sns.countplot(y = data['education'])
plt.title('Education Level Distribution')
plt.ylabel('Education Level')
plt.xlabel('Frequency')
plt.show()
```



And that's how it's done. We've just gone from a jumble of pixels to a very professional-looking plot with a few iterations. The lesson here (aside from learning about countplots) is to not be afraid to experiment! This whole process is called *exploratory* for a reason!

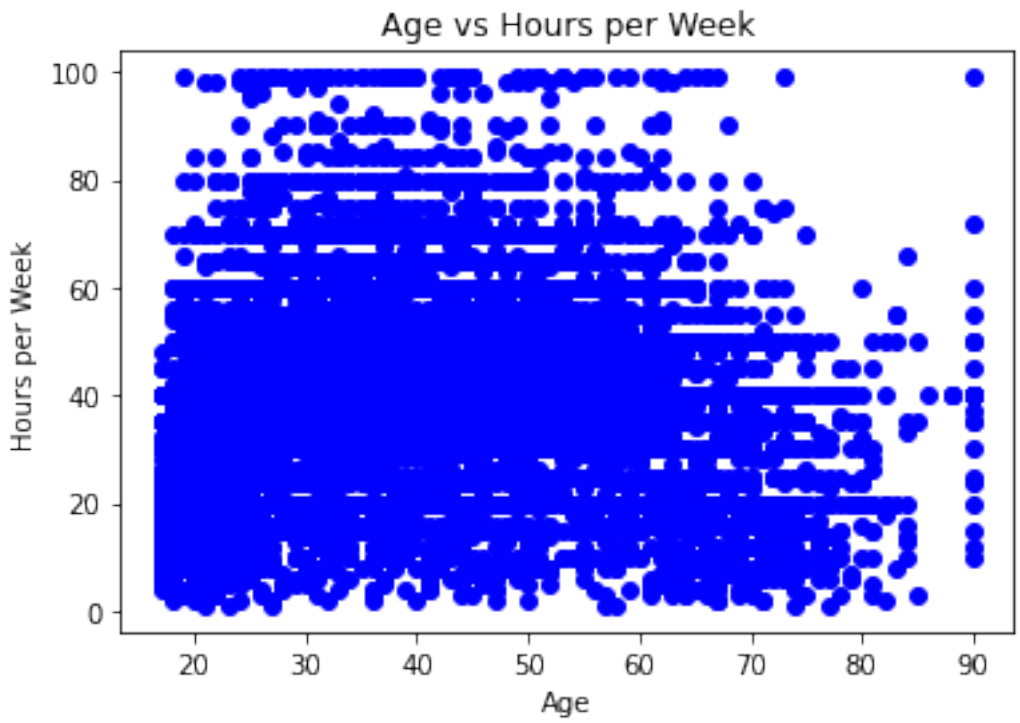
Exercise 3.2

Make another countplot corresponding to a different variable. Adjust it so it looks professional.

Two Numerical Variables: Scatterplots

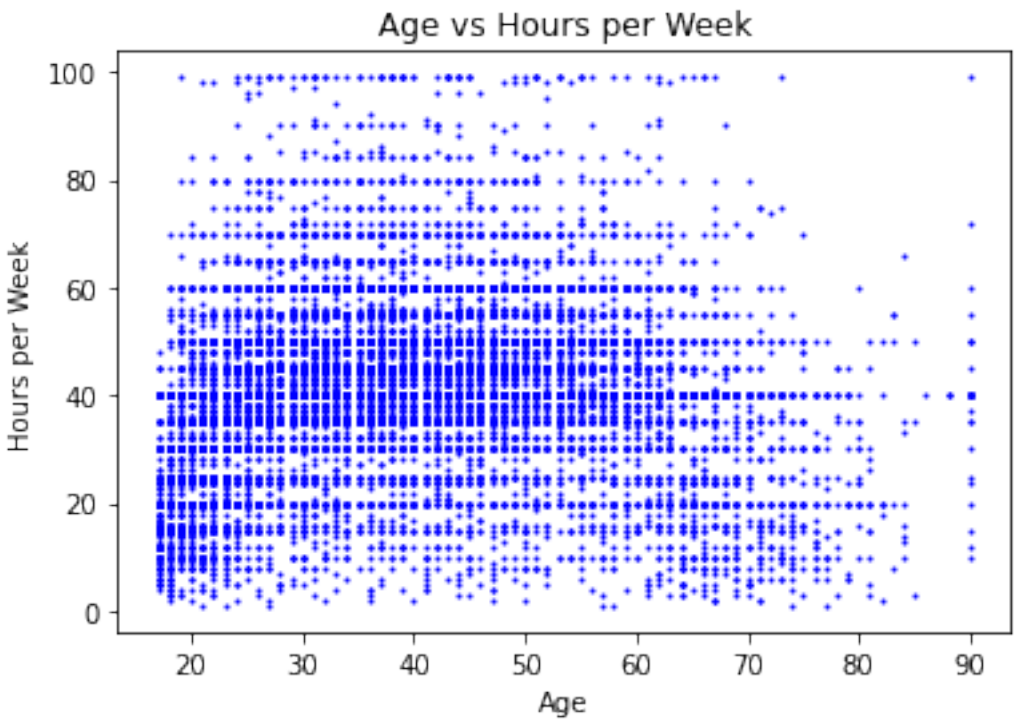
Scatterplots are so called because they scatter points across an (x, y) plane. As such, you need points with two numerical variables in order to do this. Here, let's look at `age` versus `hours-per-week`.

```
#make a scatterplot of age on the x-axis and hours-per-week on the y-axis.
plt.scatter(data['age'], data['hours-per-week'], color='blue')
plt.title('Age vs Hours per Week')
plt.xlabel('Age')
plt.ylabel('Hours per Week')
plt.show()
```



Whoa, that's a bit much. Can we reduce the size of the dots?

```
#same as before, but reduce the size of the dots
plt.scatter(data['age'], data['hours-per-week'], color='blue', s=1)
plt.title('Age vs Hours per Week')
plt.xlabel('Age')
plt.ylabel('Hours per Week')
plt.show()
```



Definitely looks a lot better.

Plots like this help you to visualize numerical relationships between variables, such as correlation. Unfortunately, it doesn't seem like there's a strong correlation here at first glance, especially if you kept the default dot size. But now, with smaller dots, we can kind of see that the number of hours per week increases slightly for adults from children then decreases for the elderly.

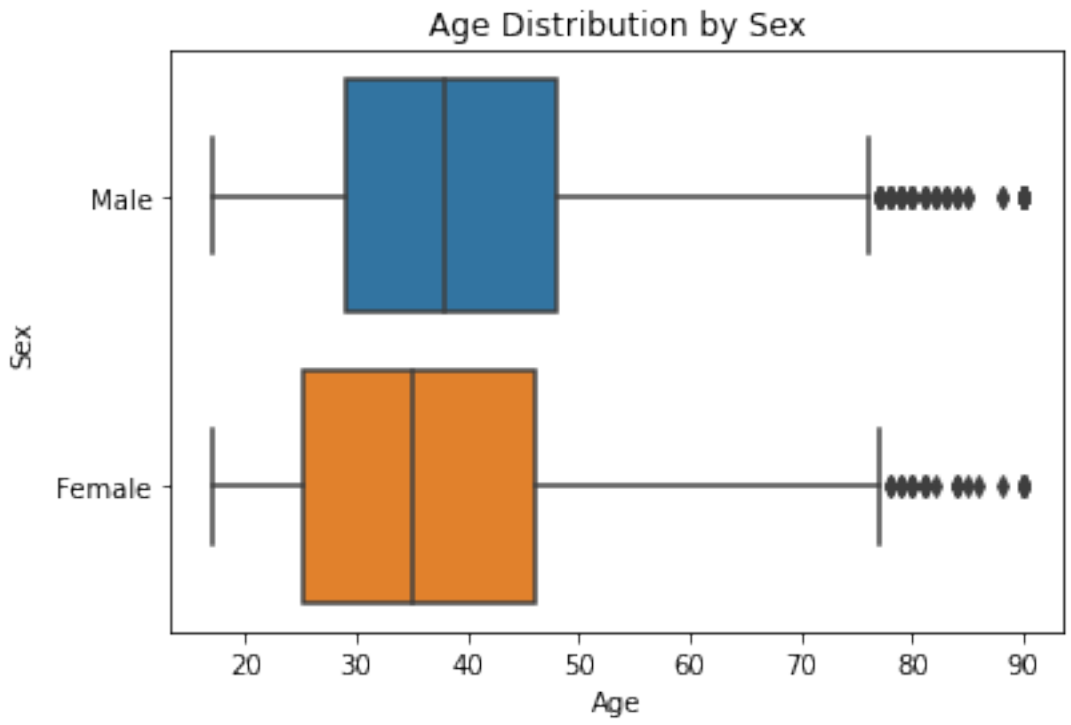
Exercise 3.3

Make another scatterplot with two other variables of your choosing. Make sure it looks presentable!

One Categorical, One Numerical: Boxplots

Boxplots, or box-and-whisker plots, help you to look at the distributions of a numerical variable when split along the categories of a categorical variable. Here, let's try it with the `age` and `sex` variables; again, we will turn to `seaborn`.

```
#use seaborn to make a boxplot of age, broken down by sex
sns.boxplot(x = data['age'], y = data['sex'])
plt.title('Age Distribution by Sex')
plt.xlabel('Age')
plt.ylabel('Sex')
plt.show()
```



Here we can see that the distributions for ages between men and women are somewhat similar. Further statistical investigation would be needed to confirm this, though.

Exercise 3.4

Make another boxplot between variables of your choosing. Make sure it looks presentable.

4.3.2 Statistical Testing

While visualizations look good and can sometimes give us information on various dataset attributes, they can be augmented by mathematical techniques to acquire more quantitative insights. These mathematical techniques are known as statistical tests; there are a variety of them and their usage depends largely on the number of variables analyzed and the distribution of the tested groups.

When performing a statistical test, we assume a *null hypothesis* H_0. Following this, the probability of the instances we have is calculated; this probability becomes a p-value. For sufficiently low p-values, we can reject the null hypothesis; this threshold is typically $p < 0.05$. In other words, getting a low enough p-value means that the null hypothesis is unlikely to be correct.

Warning

This is a massive simplification of the actual theory behind statistical testing. For more on the theory of statistical testing, refer to the Statistical Inference section.

The tests covered in this section are mainly for numerical data, and we will look at two widely used tests to compare groups: the two-sample t-test and ANOVA.

T-tests

When performing a t-test to compare two samples, you first assume the null hypothesis that the mean values for these two samples are equal; in other words, that there is no difference between the groups. You then want to acquire those two samples and then use those samples to find the p-value. The actual theory is not important here as there are many assumptions we are overlooking.

Here, let's look at the `capital-gain` variable split between sex.

```
#t-test for capital-gain between sex
from scipy import stats

#split the groups
m = data[data['sex'] == " Male"]['capital-gain']
f = data[data['sex'] == " Female"]['capital-gain']

#perform the t-test
t, p = stats.ttest_ind(m, f)
print("t-value:", t)
print("p-value:", p)
```

```
t-value: 8.487476444760196
p-value: 2.2070076201365698e-17
```

This shows that there is a significant difference between the capital gains between men and women in this dataset.

Exercise 3.5

Make plots to visualize this difference and see if the conclusion of the t-test makes sense.

Exercise 3.6

Perform another t-test in the style of your choosing. Report the t-value and p-value, and supply plots to justify this difference.

ANOVA

Analysis of Variance (more often abbreviated as ANOVA) tests are extensions of t-tests in that while t-tests compare two groups, ANOVA can compare 3 or more groups. It assumes the null hypothesis that there is no difference between the means of all of the groups; that is, there is no difference between all of the groups. Again, there is a lot of theory we are overlooking, but for practicality's sake, this is sometimes permissible.

Here, let's split the `capital-gains` variable across the `education` category.

```
#anova for capital-gains split by education
import statsmodels.api as sm
from statsmodels.formula.api import ols

#there is an error in the data that will cause the anova to fail
#capital-gain needs to be replaced with capital_gain
#otherwise the ANOVA will read this as a minus sign
data.rename(columns={'capital-gain':'capital_gain'}, inplace=True)
```

(continues on next page)

(continued from previous page)

```
data.head()

#perform the anova
model = ols('capital_gain ~ education', data=data).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
print(anova_table)
```

```
                sum_sq       df          F         PR(>F)
education  6.886420e+10     15.0  87.285486  4.079945e-264
Residual   1.585586e+12  30146.0        NaN            NaN
```

Our ANOVA result outputs a table. For basic purposes, the most important result is the one in the last column, the PR(>F) value. This is a very low p-value, so preliminarily we would say that there are significant differences between capital gains by education levels.

Exercise 3.7

Does the test tell us where the significant difference lies?

Exercise 3.8

Perform another ANOVA test of your choosing.

4.3.3 EDA Project

Find another dataset of your choosing on the internet and perform some of the techniques described in this basic EDA part of the chapter. Make sure your results are presentable and that you add some textual description to them.

Common places to find data include:

- UC Irvine Machine Learning Repository, https://archive.ics.uci.edu/
- Kaggle, https://www.kaggle.com/
- The GitHub Repository of Awesome Public Datasets, https://github.com/awesomedata/awesome-public-datasets
- KDNuggets, which links many more data sources: https://www.kdnuggets.com/datasets/index.html

4.4 Solutions to Part I Exercises

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Load the data
data = pd.read_csv('census_income.csv')
```

Exercise 2.1

```
data.tail()
```

```
       age      workclass  fnlwgt    education  education-num  \
32556   27        Private  257302   Assoc-acdm             12
32557   40        Private  154374      HS-grad              9
32558   58        Private  151910      HS-grad              9
32559   22        Private  201490      HS-grad              9
32560   52   Self-emp-inc  287927      HS-grad              9

           marital-status          occupation relationship    race      sex  \
32556   Married-civ-spouse        Tech-support         Wife   White   Female
32557   Married-civ-spouse   Machine-op-inspct      Husband   White     Male
32558              Widowed        Adm-clerical    Unmarried   White   Female
32559        Never-married        Adm-clerical    Own-child   White     Male
32560   Married-civ-spouse     Exec-managerial         Wife   White   Female

       capital-gain  capital-loss  hours-per-week  native-country  income
32556             0             0              38   United-States   <=50K
32557             0             0              40   United-States    >50K
32558             0             0              40   United-States   <=50K
32559             0             0              20   United-States   <=50K
32560         15024             0              40   United-States    >50K
```

Exercise 2.2: Answers will vary depending on what students hypothesize each variable to be.

Exercise 2.3:

```
#remove N/A rows
data = data.dropna()
data.describe()
```

```
                age        fnlwgt  education-num  capital-gain  capital-loss  \
count  30162.000000  3.016200e+04   30162.000000  30162.000000  30162.000000
mean      38.437902  1.897938e+05      10.121312   1092.007858     88.372489
std       13.134665  1.056530e+05       2.549995   7406.346497    404.298370
min       17.000000  1.376900e+04       1.000000      0.000000      0.000000
25%       28.000000  1.176272e+05       9.000000      0.000000      0.000000
50%       37.000000  1.784250e+05      10.000000      0.000000      0.000000
75%       47.000000  2.376285e+05      13.000000      0.000000      0.000000
max       90.000000  1.484705e+06      16.000000  99999.000000   4356.000000

       hours-per-week
count    30162.000000
mean        40.931238
std         11.979984
min          1.000000
25%         40.000000
50%         40.000000
75%         45.000000
max         99.000000
```

Since we dropped a small proportion of rows, we should not see too many changes in the summary statistics.

Exercise 2.4:

```
#for the education column
#print out the categories in the column
#then the number of instances for each category
print(data['education'].value_counts())
```

```
 HS-grad         9840
 Some-college    6678
 Bachelors       5044
 Masters         1627
 Assoc-voc       1307
 11th            1048
 Assoc-acdm      1008
 10th             820
 7th-8th          557
 Prof-school      542
 9th              455
 12th             377
 Doctorate        375
 5th-6th          288
 1st-4th          151
 Preschool         45
Name: education, dtype: int64
```

Exercise 2.5:

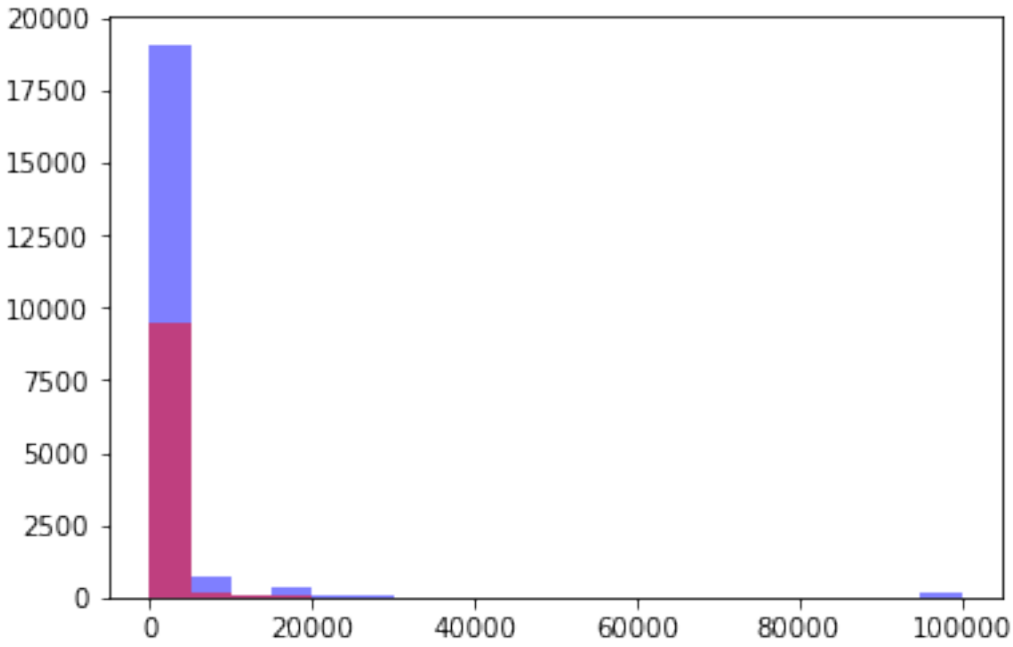
```
#function get_value_counts that takes in a variable name (string)
#and returns the value counts for that
def get_value_counts(var_name):
    return data[var_name].value_counts()

#test
print(get_value_counts('occupation'))
```

```
 Prof-specialty       4038
 Craft-repair         4030
 Exec-managerial      3992
 Adm-clerical         3721
 Sales                3584
 Other-service        3212
 Machine-op-inspct    1966
 Transport-moving     1572
 Handlers-cleaners    1350
 Farming-fishing       989
 Tech-support          912
 Protective-serv       644
 Priv-house-serv       143
 Armed-Forces            9
Name: occupation, dtype: int64
```

Exercises 3.1-3.4: Answers will vary depending on the chosen variable, but the code should follow the examples given in the section and the plots should be readable.

Exercise 3.5:

```
#split capital-gain by sex
m = data[data['sex'] == " Male"]['capital-gain']
f = data[data['sex'] == " Female"]['capital-gain']

#we are going to plot the histograms of the capital
#with a different color for each sex
plt.hist(m, bins=20, alpha=0.5, color='b')
plt.hist(f, bins=20, alpha=0.5, color='r')
plt.show()
```



Interpretation of this may vary. An example is that the difference seems to largely be due to one single outlier male with lots of capital gain.

Exercise 3.6: Answers may vary depending on the variables chosen but should follow the example code and the plotting above.

Exercise 3.7: No, the test shows us that there is a difference, but it does not tell us where that difference lies.

Exercise 3.8: Answers may vary depending on the variables chosen but should follow the example code.

4.5 Introduction to Part II

In this portion of the chapter, we will introduce some more challenging methods for doing exploratory data analysis. This portion can be taken on its own or in conjunction with the previous sections on basic EDA.

This chapter will go over the following:

- Basic data examination
- Removing N/A values
- Basic transformations
- Advanced transformations
- Plotting with `matplotlib` and `seaborn`
- Statistical testing
- Writing and producing reports

Here, we will aim to focus on the practical methodologies of implementing these procedures, and much less so the mathematical theory behind how and why they work. Additionally, it may be more open-ended in terms of exercises and may require readers to consult outside resources, such as Python package documentation.

4.5.1 About the Dataset

The dataset used for the advanced EDA portion of this chapter comes from the Department of Human Services (DHS) in Allegheny County, Pennsylvania, USA, where the city of Pittsburgh is located. These data are a synthetic version of data used by the DHS for recordkeeping of the usage of their services; these records have been used in projects utilising machine learning to assess child maltreatment risk (see https://proceedings.mlr.press/v81/chouldechova18a.html). Since the raw version of the records data contains large amounts of sensitive and private information, the DHS, along with CountyStat, Pittsburgh's Urban Institute, and the University of Pittsburgh's Western Pennsylvania Regional Data Center (WPRDC) created a synthetic version that does not carry sensitive information and is therefore freely usable to the public.

The data can be found at https://data.wprdc.org/dataset/synthetic-integrated-services-data, along with other information about the dataset and synthetic data in general. However, since it is a large dataset and may take a while to download, we recommend you download the compressed version at https://data.wprdc.org/dataset/synthetic-integrated-services-data/resource/d5b12f18-ce4e-4565-b7ed-d5565e18bd43. More information on the data will be discovered in later sections.

4.5.2 What is EDA?

EDA, or exploratory data analysis, is the process of looking at, cleaning, transforming, and making small insights about data before it is used in more advanced methodologies such as machine learning.

The process of EDA varies between datasets and does not have a specific procedure. However, it typically follows the following process:

1. Looking at the data
2. Cleaning the data
3. Transforming the data
4. Getting insights from data

Data scientists like to describe EDA as the process of "having conversations" with the data. As you have conversations with real-life people, you find out more and more about them. The first time you talk to someone new, you may only find out things about their day-to-day life or some of their hobbies. However, with more conversations, you get to know them better, and that knowledge can reveal even more about the person with more conversations. This metaphor is very much the same when working with data - the more you work with it, look at it, and experiment with it, the more you will learn about the data.

4.5.3 Why is EDA Important?

Very few times in academia or industry will you encounter a dataset that is ready-made and good to go for advanced uses like machine learning. In fact, even if you have such a dataset, it is often a bad idea to jump straight into advanced uses of the data and coming to conclusions about it without looking at it first. Real-world data is often full of missing values, incomplete entries, and improper formatting, and going into usage of the data without understanding it first will result in these errors influencing any conclusions you make about the data.

A majority of the time spent in EDA is spent cleaning up the data and making it good for use. While humans may be able to read small datasets and perform operations on them just fine, computers are often more finicky and perform differently under different circumstances. For example, one missing value will result in a computer not being able to compute the mean of a set of numbers; in another case, having a variable set as the "string" datatype instead of "category" may make the computer run more slowly as it treats these differently.

4.5.4 Integrating Domain Knowledge

Every data analysis setting has a context in which it lies, whether you are doing the analysis for a course project or for your job. In any case, data analysis does not occur in a bubble.

Data analysis needs the input of those who are knowledgeable about the context of the data; these people are called *domain experts*, and their knowledge is called *domain knowledge*. Data analysis should build on what domain experts already know; in turn, domain knowledge can increase with subsequent data analysis. Without each other, progress towards understanding the data will be slow, if existent at all.

4.5.5 Guiding Questions

When performing EDA, we often have guiding questions that will help us make a straightforward path for our analysis. These questions can come from domain experts, papers that have used the data before, or in some cases even thinking about the data itself (again, more conversations with the data can open up avenues to get to know the data even better).

4.5.6 What Makes EDA Here "Advanced"?

While the EDA here builds on the basics of EDA covered in previous sections, the EDA here does not simply utilize pre-existing Python packages. In some cases, it may rely on writing functions to accomplish a specific goal; this involves a lot of logical thinking about how to accomplish this goal and code it.

Another reason EDA in this section is more advanced is that there are far fewer exercises, replaced instead with a section-spanning project that concerns this whole dataset. This project may also require the use of outside resources and problem-solving skills that cannot be succinctly explained in this chapter. Because of this, the project will require more knowledge *and* take more time to complete.

It should also be noted that using Python to conduct EDA is just one way of doing EDA. There are a myriad of tools that each have their own distinct uses, each of which can be useful for EDA. Some tools include Tableau, a commonly used software used for data visualization without extensive coding; even Excel can be useful for looking at datasets, though its capabilities are far less than if you were to use a programming language. Additionally, since the datasets we are using so far are very big, other programming languages can be helpful.

So far, this entire book has used Python, which is known as an *interpreter language*; every line you write in Python, when run, makes subsequent calls to built-in lines of code, whether in Python or in another language like C++. This makes many layers upon layers of code calling other lines of code. As such, Python is often memory-intensive and may take longer to run than interpreter languages' counterparts, known as *compiler languages*. With compiler languages, the code you write is directly translated into machine code for the computer to execute. This means far fewer layers of called lines of code and the procedure becomes less memory intensive.

The downside of many compiler languages is that they are semantically farther away from normal human language (in this case English) than interpreter languages like Python. As such, they are more advanced to read and understand what is going on. Additionally, since there are not as many widely-used purpose-built packages for compiler languages, it often means having to write more code that is less flexible (e.g., certain languages cannot handle datasets with both integers and strings).

Ultimately it is up to you what tools to use. In this section, we will only be using Python, but knowing that you can use whatever tools are most comfortable to you may be helpful in your future endeavors and in the project presented in this chapter.

4.6 Data Investigation II

Often times when we explore data for the first time, we have to spend time looking at what is missing in the data, or what needs to be changed. Data files rarely if ever are perfect for use immediately once you get them. Additionally, it is often unwise to utilize data for predictive modeling or statistical analysis without knowing some of the context for the data and looking at its contents.

Python has a helpful package called `pandas` which is purpose-built for looking at and manipulating data, and a lot of what we will work on in this section will involve using it.

4.6.1 Setup

The first thing we typically do when dealing with data in Python is to import the `pandas` package and one of its dependencies, `numpy`.

```
import pandas as pd
import numpy as np
```

Due to the large size of the provided data file, we store it in a different directory than the one where this notebook lies. We can easily set this directory using the `os` package.

```
import os
os.chdir('<your-directory>')
df = pd.read_csv('dhs_service_records_synthesized_final.csv')
```

4.6.2 Simple Data Inspection

Before doing anything with the data, it is good practice to take a look at a few samples from the dataset. Here, we're going to take a look at the first ten rows and the last seven rows, or the head and tail of the data; this is a function that is easy to call, and it is good practice to call either of these often so that you know how a dataset evolves over time.

```
df.head()
```

```
   synthetic_data  MCI_UNIQ_ID  CALDR_YR DATE_OF_EVENT  \
0  SYNTHETIC DATA         1626      2021    01/31/2021
1  SYNTHETIC DATA         1626      2021    02/28/2021
2  SYNTHETIC DATA         1626      2021    03/31/2021
3  SYNTHETIC DATA         1626      2021    04/30/2021
4  SYNTHETIC DATA         1626      2021    05/31/2021

                                          service  GEO_AREA  age  DOB  DOD  \
0  Children_Attending_Early_Childhood_Programs_Ma...       NaN    5  NaN  NaN
1  Children_Attending_Early_Childhood_Programs_Ma...       NaN    5  NaN  NaN
2  Children_Attending_Early_Childhood_Programs_Ma...       NaN    5  NaN  NaN
3  Children_Attending_Early_Childhood_Programs_Ma...       NaN    5  NaN  NaN
4  Children_Attending_Early_Childhood_Programs_Ma...       NaN    5  NaN  NaN

   GENDER  GENDER_IDENTITY  SEX_ORIENT  LEGAL_SEX     RACE   ETHNICITY  \
0  1~Male              NaN         NaN        NaN  1~White  99~Unknown
1  1~Male              NaN         NaN        NaN  1~White  99~Unknown
2  1~Male              NaN         NaN        NaN  1~White  99~Unknown
3  1~Male              NaN         NaN        NaN  1~White  99~Unknown
4  1~Male              NaN         NaN        NaN  1~White  99~Unknown
```

(continues on next page)

(continued from previous page)

```
   LIVING_ARRANGEMENT  EMPLOYMENT_STATUS MARITAL_STATUS  \
0                 NaN               NaN     99~Unknown
1                 NaN               NaN     99~Unknown
2                 NaN               NaN     99~Unknown
3                 NaN               NaN     99~Unknown
4                 NaN               NaN     99~Unknown

                EDUCATION_LEVEL  VETERAN_FLAG
0  9-12~High School (grade 9-12)           NaN
1  9-12~High School (grade 9-12)           NaN
2  9-12~High School (grade 9-12)           NaN
3  9-12~High School (grade 9-12)           NaN
4  9-12~High School (grade 9-12)           NaN
```

```
df.tail(7)
```

```
         synthetic_data  MCI_UNIQ_ID  CALDR_YR DATE_OF_EVENT   service  \
7116127  SYNTHETIC DATA       530696      2021    06/30/2021  Suicides
7116128  SYNTHETIC DATA       530765      2021    06/30/2021  Suicides
7116129  SYNTHETIC DATA       530832      2021    06/30/2021  Suicides
7116130  SYNTHETIC DATA       532121      2021    04/30/2021  Suicides
7116131  SYNTHETIC DATA       533991      2021    11/30/2021  Suicides
7116132  SYNTHETIC DATA       534127      2021    07/31/2021  Suicides
7116133  SYNTHETIC DATA       535156      2021    06/30/2021  Suicides

         GEO_AREA  age  DOB  DOD    GENDER  GENDER_IDENTITY  SEX_ORIENT  \
7116127       NaN   21  NaN  NaN  2~Female              NaN         NaN
7116128       NaN   15  NaN  NaN  2~Female              NaN         NaN
7116129       NaN   17  NaN  NaN    1~Male              NaN         NaN
7116130       NaN   22  NaN  NaN  2~Female              NaN         NaN
7116131       NaN   16  NaN  NaN    1~Male              NaN         NaN
7116132       NaN   19  NaN  NaN    1~Male              NaN         NaN
7116133       NaN   24  NaN  NaN    1~Male              NaN         NaN

         LEGAL_SEX     RACE              ETHNICITY  LIVING_ARRANGEMENT  \
7116127       NaN  1~White             99~Unknown                 NaN
7116128       NaN  1~White             99~Unknown                 NaN
7116129       NaN  1~White             99~Unknown                 NaN
7116130       NaN  1~White             99~Unknown                 NaN
7116131       NaN  1~White             99~Unknown                 NaN
7116132       NaN  1~White  2~Not Hispanic/Latinx                 NaN
7116133       NaN  1~White             99~Unknown                 NaN

         EMPLOYMENT_STATUS          MARITAL_STATUS  \
7116127               NaN             99~Unknown
7116128               NaN             99~Unknown
7116129               NaN             99~Unknown
7116130               NaN  1~Single-Never Married
7116131               NaN  1~Single-Never Married
7116132               NaN               4~Widowed
7116133               NaN  1~Single-Never Married

                    EDUCATION_LEVEL  VETERAN_FLAG
7116127                 99~Unknown           NaN
7116128                 99~Unknown           NaN
```

(continues on next page)

(continued from previous page)

```
7116129                    99~Unknown          NaN
7116130  9-12~High School (grade 9-12)          NaN
7116131  9-12~High School (grade 9-12)          NaN
7116132           GRAD~Graduate Degree          NaN
7116133  9-12~High School (grade 9-12)          NaN
```

4.6.3 Dataframe Attributes

You can get a lot of helpful information from the attributes of the dataframe; specifically, we are going to look at dataframe dimensions, column names, column data types, and the ranges of column values. To get these, we can easily call some prebuilt attributes that are assigned to pandas dataframes. These help us to understand our data before we even start modifying it.

```
df.shape
```

```
(7116134, 20)
```

```
df.columns
```

```
Index(['synthetic_data', 'MCI_UNIQ_ID', 'CALDR_YR', 'DATE_OF_EVENT', 'service',
       'GEO_AREA', 'age', 'DOB', 'DOD', 'GENDER', 'GENDER_IDENTITY',
       'SEX_ORIENT', 'LEGAL_SEX', 'RACE', 'ETHNICITY', 'LIVING_ARRANGEMENT',
       'EMPLOYMENT_STATUS', 'MARITAL_STATUS', 'EDUCATION_LEVEL',
       'VETERAN_FLAG'],
      dtype='object')
```

Pandas dataframes also have a helpful method for easy summarization of the set, which you can call with `df.info()`.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7116134 entries, 0 to 7116133
Data columns (total 20 columns):
 #   Column              Dtype
---  ------              -----
 0   synthetic_data      object
 1   MCI_UNIQ_ID         int64
 2   CALDR_YR            int64
 3   DATE_OF_EVENT       object
 4   service             object
 5   GEO_AREA            float64
 6   age                 int64
 7   DOB                 float64
 8   DOD                 float64
 9   GENDER              object
 10  GENDER_IDENTITY     float64
 11  SEX_ORIENT          float64
 12  LEGAL_SEX           float64
 13  RACE                object
 14  ETHNICITY           object
 15  LIVING_ARRANGEMENT  float64
 16  EMPLOYMENT_STATUS   float64
 17  MARITAL_STATUS      object
```

(continues on next page)

(continued from previous page)

```
 18  EDUCATION_LEVEL     object
 19  VETERAN_FLAG        float64
dtypes: float64(9), int64(3), object(8)
memory usage: 1.1+ GB
```

Another way to check the data types, if you just want that, is to use the `dtypes` attribute of the dataframe.

```
df.dtypes
```

```
synthetic_data         object
MCI_UNIQ_ID             int64
CALDR_YR                int64
DATE_OF_EVENT          object
service                object
GEO_AREA              float64
age                     int64
DOB                   float64
DOD                   float64
GENDER                 object
GENDER_IDENTITY       float64
SEX_ORIENT            float64
LEGAL_SEX             float64
RACE                   object
ETHNICITY              object
LIVING_ARRANGEMENT    float64
EMPLOYMENT_STATUS     float64
MARITAL_STATUS         object
EDUCATION_LEVEL        object
VETERAN_FLAG          float64
dtype: object
```

We can also find the number of unique entries in the dataframe. This can be important as in some cases we may not want duplicates. In the case of these data from Allegheny County, unique IDs are important as each one will correspond to a single person, and we can track that person across their usage of many different services.

```
df.nunique()
```

```
synthetic_data             1
MCI_UNIQ_ID           533799
CALDR_YR                   1
DATE_OF_EVENT             12
service                   22
GEO_AREA                   0
age                      112
DOB                        0
DOD                        0
GENDER                     3
GENDER_IDENTITY            0
SEX_ORIENT                 0
LEGAL_SEX                  0
RACE                       7
ETHNICITY                  3
LIVING_ARRANGEMENT         0
EMPLOYMENT_STATUS          0
MARITAL_STATUS             7
EDUCATION_LEVEL            8
```

(continues on next page)

(continued from previous page)

```
VETERAN_FLAG               0
dtype: int64
```

Note to collaborators: the next two code cells are irrelevant with the complete dataset as there are no entries with "SYNTHET" in the synthetic_data column.

```
#note: I am not sure what 'SYNTHET" versus "SYNTHETIC DATA" means.
#but we split by them anyway.
df.groupby("synthetic_data").count()
```

```
                MCI_UNIQ_ID  CALDR_YR  DATE_OF_EVENT  service  GEO_AREA  \
synthetic_data
SYNTHETIC DATA      7116134   7116134        7116134  7116134         0

                    age  DOB  DOD   GENDER  GENDER_IDENTITY  SEX_ORIENT  \
synthetic_data
SYNTHETIC DATA  7116134    0    0  7116134                0           0

                LEGAL_SEX     RACE  ETHNICITY  LIVING_ARRANGEMENT  \
synthetic_data
SYNTHETIC DATA          0  7116134    7116134                   0

                EMPLOYMENT_STATUS  MARITAL_STATUS  EDUCATION_LEVEL  \
synthetic_data
SYNTHETIC DATA                  0         7116134          7116134

                VETERAN_FLAG
synthetic_data
SYNTHETIC DATA             0
```

```
df["synthetic_data"].value_counts()
```

```
SYNTHETIC DATA    7116134
Name: synthetic_data, dtype: int64
```

One way to get an overview of the data values is to use another helpful function, `df.describe()`. This will provide several summary statistics for you for each of the columns in the set.

```
df.describe()
```

```
        MCI_UNIQ_ID   CALDR_YR  GEO_AREA           age  DOB  DOD  \
count  7.116134e+06  7116134.0       0.0  7.116134e+06  0.0  0.0
mean   2.674853e+05     2021.0       NaN  3.267702e+01  NaN  NaN
std    1.544397e+05        0.0       NaN  2.258095e+01  NaN  NaN
min    1.000000e+00     2021.0       NaN  0.000000e+00  NaN  NaN
25%    1.337730e+05     2021.0       NaN  1.300000e+01  NaN  NaN
50%    2.675410e+05     2021.0       NaN  3.000000e+01  NaN  NaN
75%    4.010820e+05     2021.0       NaN  4.900000e+01  NaN  NaN
max    5.356080e+05     2021.0       NaN  1.210000e+02  NaN  NaN

       GENDER_IDENTITY  SEX_ORIENT  LEGAL_SEX  LIVING_ARRANGEMENT  \
count              0.0         0.0        0.0                 0.0
mean               NaN         NaN        NaN                 NaN
std                NaN         NaN        NaN                 NaN
min                NaN         NaN        NaN                 NaN
```

(continues on next page)

(continued from previous page)

```
25%                NaN         NaN        NaN                  NaN
50%                NaN         NaN        NaN                  NaN
75%                NaN         NaN        NaN                  NaN
max                NaN         NaN        NaN                  NaN

       EMPLOYMENT_STATUS  VETERAN_FLAG
count                0.0           0.0
mean                 NaN           NaN
std                  NaN           NaN
min                  NaN           NaN
25%                  NaN           NaN
50%                  NaN           NaN
75%                  NaN           NaN
max                  NaN           NaN
```

In some cases, these are not helpful, like in the IDs portion, having the mean ID value is not important. Same with calendar year, especially since these are all from the same year. But something to note is that the values for the mean, std error, and quartiles will not show up if there are ANY NaN values in the column. So we need to find out where these NaN values are.

4.6.4 Finding NaN values

To get the number of NaN (N/A or NA, as they are sometimes called), we can use the `isna()` method in combination with the `sum()` method.

```
df.isna().sum()
```

```
synthetic_data              0
MCI_UNIQ_ID                 0
CALDR_YR                    0
DATE_OF_EVENT               0
service                     0
GEO_AREA              7116134
age                         0
DOB                   7116134
DOD                   7116134
GENDER                      0
GENDER_IDENTITY       7116134
SEX_ORIENT            7116134
LEGAL_SEX             7116134
RACE                        0
ETHNICITY                   0
LIVING_ARRANGEMENT    7116134
EMPLOYMENT_STATUS     7116134
MARITAL_STATUS              0
EDUCATION_LEVEL             0
VETERAN_FLAG          7116134
dtype: int64
```

Note for collaborators: df.isna() and df.isnull() now produce the same results, so the below code cell is not relevant.

```
df.isnull().sum()
```

```
synthetic_data              0
MCI_UNIQ_ID                 0
CALDR_YR                    0
DATE_OF_EVENT               0
service                     0
GEO_AREA              7116134
age                         0
DOB                   7116134
DOD                   7116134
GENDER                      0
GENDER_IDENTITY       7116134
SEX_ORIENT            7116134
LEGAL_SEX             7116134
RACE                        0
ETHNICITY                   0
LIVING_ARRANGEMENT    7116134
EMPLOYMENT_STATUS     7116134
MARITAL_STATUS              0
EDUCATION_LEVEL             0
VETERAN_FLAG          7116134
dtype: int64
```

We can see the columns [‘GEO_AREA’, ‘DOB’, ‘DOD’, ‘GENDER_IDENTITY’, ‘SEX_ORIENT’, ‘LEGAL_SEX’, ‘LIVING_ARRANGEMENT’, ‘EMPLOYMENT_STATUS’, ‘VETERAN_FLAG’] have only NaN values.

Note to collaborators: the bottom row that has only NAs is gone with the complete version of the data.

```
df.isna().tail(1)
```

```
         synthetic_data  MCI_UNIQ_ID  CALDR_YR  DATE_OF_EVENT  service  \
7116133           False        False     False          False    False

         GEO_AREA    age   DOB   DOD  GENDER  GENDER_IDENTITY  SEX_ORIENT  \
7116133      True  False  True  True   False             True        True

         LEGAL_SEX   RACE  ETHNICITY  LIVING_ARRANGEMENT  EMPLOYMENT_STATUS  \
7116133       True  False      False                True               True

         MARITAL_STATUS  EDUCATION_LEVEL  VETERAN_FLAG
7116133           False            False          True
```

let’s clean up the data frame then, drop last row, drop the columns that only have NAs

Note: I am also not quite sure what the code cell above and below are doing in this context.

```
df.tail(1)
```

```
         synthetic_data  MCI_UNIQ_ID  CALDR_YR DATE_OF_EVENT   service  \
7116133  SYNTHETIC DATA       535156      2021    06/30/2021  Suicides

         GEO_AREA  age  DOB  DOD  GENDER  GENDER_IDENTITY  SEX_ORIENT  \
7116133       NaN   24  NaN  NaN  1~Male              NaN         NaN

         LEGAL_SEX     RACE   ETHNICITY  LIVING_ARRANGEMENT  \
7116133        NaN  1~White  99~Unknown                 NaN

         EMPLOYMENT_STATUS          MARITAL_STATUS  \
```

(continues on next page)

(continued from previous page)

```
7116133                NaN  1~Single-Never Married

                      EDUCATION_LEVEL  VETERAN_FLAG
7116133  9-12~High School (grade 9-12)           NaN
```

4.6.5 Removing NA Columns and Rows

To drop all rows with only NaN values, we can use a simple function called `df.dropna()`.

```
df.dropna(how='all', inplace=True)
df.shape
```

```
(7116134, 20)
```

With this function, the default parameter for a hidden value `axis` removed rows of only NA values, of which there are none in the dataset. To remove columns, we need to specify the axis.

```
#we want to remove all the columns that only have NA values
df.dropna(axis=1, how='all', inplace=True)
df.shape
```

```
(7116134, 11)
```

Now what if we wanted to drop a row with a single NA value? We can simply remove the how='all' parameter and it will remove the rows with at least one NA value.

```
# df.dropna(inplace=True) # will drop rows with any NA.
```

In addition to looking at the head and the tail, we are also at this point going to take a look at 3 random rows, as it is important to inspect data beyond the head and tail.

```
# set seed for random function so that we get same rows when re-run the cell
np.random.seed(17)
np.random.randint(2, df.shape[0],3)
```

```
array([ 589043, 5539985, 2078088])
```

```
np.random.randint(3, df.shape[0],3)
```

```
array([ 491929,  828732, 6417139])
```

```
# set seed for random function so that we get same rows when re-run the cell
np.random.seed(17)
df.iloc[np.random.randint(0, df.shape[0],3)]
```

```
         synthetic_data  MCI_UNIQ_ID  CALDR_YR DATE_OF_EVENT  \
589041   SYNTHETIC DATA       364757      2021    10/31/2021
5539983  SYNTHETIC DATA       449747      2021    09/30/2021
2078086  SYNTHETIC DATA       109525      2021    09/30/2021

                                    service  age  GENDER     RACE  \
589041      Individuals_Receiving_DHS_Services   54  1~Male  1~White
```

(continues on next page)

(continued from previous page)

```
5539983  Individuals_Receiving_Income_Supports   51  1~Male  1~White
2078086  Individuals_Receiving_Income_Supports   51  1~Male  1~White

                    ETHNICITY          MARITAL_STATUS EDUCATION_LEVEL
589041   2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
5539983  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
2078086  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
```

We will also take a look at the record for a specific MCI_UNIQ_ID. Let's look at client 364757.

```
df[df['MCI_UNIQ_ID'] == 364757]
```

```
         synthetic_data  MCI_UNIQ_ID  CALDR_YR DATE_OF_EVENT  \
589034   SYNTHETIC DATA       364757      2021    02/28/2021
589035   SYNTHETIC DATA       364757      2021    03/31/2021
589036   SYNTHETIC DATA       364757      2021    05/31/2021
589037   SYNTHETIC DATA       364757      2021    06/30/2021
589038   SYNTHETIC DATA       364757      2021    07/31/2021
589039   SYNTHETIC DATA       364757      2021    08/31/2021
589040   SYNTHETIC DATA       364757      2021    09/30/2021
589041   SYNTHETIC DATA       364757      2021    10/31/2021
589042   SYNTHETIC DATA       364757      2021    11/30/2021
589043   SYNTHETIC DATA       364757      2021    12/31/2021
4675581  SYNTHETIC DATA       364757      2021    01/31/2021
4675582  SYNTHETIC DATA       364757      2021    02/28/2021
4675583  SYNTHETIC DATA       364757      2021    03/31/2021
4675584  SYNTHETIC DATA       364757      2021    04/30/2021
4675585  SYNTHETIC DATA       364757      2021    05/31/2021
4675586  SYNTHETIC DATA       364757      2021    06/30/2021
4675587  SYNTHETIC DATA       364757      2021    07/31/2021
4675588  SYNTHETIC DATA       364757      2021    08/31/2021
4675589  SYNTHETIC DATA       364757      2021    09/30/2021
4675590  SYNTHETIC DATA       364757      2021    10/31/2021
4675591  SYNTHETIC DATA       364757      2021    11/30/2021
4675592  SYNTHETIC DATA       364757      2021    12/31/2021
6710438  SYNTHETIC DATA       364757      2021    01/31/2021
6710439  SYNTHETIC DATA       364757      2021    03/31/2021
6710440  SYNTHETIC DATA       364757      2021    04/30/2021
6710441  SYNTHETIC DATA       364757      2021    05/31/2021
6710442  SYNTHETIC DATA       364757      2021    06/30/2021
6710443  SYNTHETIC DATA       364757      2021    07/31/2021
6710444  SYNTHETIC DATA       364757      2021    08/31/2021
6710445  SYNTHETIC DATA       364757      2021    10/31/2021
6710446  SYNTHETIC DATA       364757      2021    11/30/2021
6710447  SYNTHETIC DATA       364757      2021    12/31/2021

                                          service  age  GENDER     RACE  \
589034             Individuals_Receiving_DHS_Services   54  1~Male  1~White
589035             Individuals_Receiving_DHS_Services   54  1~Male  1~White
589036             Individuals_Receiving_DHS_Services   54  1~Male  1~White
589037             Individuals_Receiving_DHS_Services   54  1~Male  1~White
589038             Individuals_Receiving_DHS_Services   54  1~Male  1~White
589039             Individuals_Receiving_DHS_Services   54  1~Male  1~White
589040             Individuals_Receiving_DHS_Services   54  1~Male  1~White
589041             Individuals_Receiving_DHS_Services   54  1~Male  1~White
589042             Individuals_Receiving_DHS_Services   54  1~Male  1~White
```

(continues on next page)

(continued from previous page)

```
589043             Individuals_Receiving_DHS_Services   54  1~Male  1~White
4675581         Individuals_Receiving_Income_Supports   54  1~Male  1~White
4675582         Individuals_Receiving_Income_Supports   54  1~Male  1~White
4675583         Individuals_Receiving_Income_Supports   54  1~Male  1~White
4675584         Individuals_Receiving_Income_Supports   54  1~Male  1~White
4675585         Individuals_Receiving_Income_Supports   54  1~Male  1~White
4675586         Individuals_Receiving_Income_Supports   54  1~Male  1~White
4675587         Individuals_Receiving_Income_Supports   54  1~Male  1~White
4675588         Individuals_Receiving_Income_Supports   54  1~Male  1~White
4675589         Individuals_Receiving_Income_Supports   54  1~Male  1~White
4675590         Individuals_Receiving_Income_Supports   54  1~Male  1~White
4675591         Individuals_Receiving_Income_Supports   54  1~Male  1~White
4675592         Individuals_Receiving_Income_Supports   54  1~Male  1~White
6710438  Individuals_Receiving_Mental_Health_Services   54  1~Male  1~White
6710439  Individuals_Receiving_Mental_Health_Services   54  1~Male  1~White
6710440  Individuals_Receiving_Mental_Health_Services   54  1~Male  1~White
6710441  Individuals_Receiving_Mental_Health_Services   54  1~Male  1~White
6710442  Individuals_Receiving_Mental_Health_Services   54  1~Male  1~White
6710443  Individuals_Receiving_Mental_Health_Services   54  1~Male  1~White
6710444  Individuals_Receiving_Mental_Health_Services   54  1~Male  1~White
6710445  Individuals_Receiving_Mental_Health_Services   54  1~Male  1~White
6710446  Individuals_Receiving_Mental_Health_Services   54  1~Male  1~White
6710447  Individuals_Receiving_Mental_Health_Services   54  1~Male  1~White

                    ETHNICITY          MARITAL_STATUS EDUCATION_LEVEL
589034   2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
589035   2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
589036   2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
589037   2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
589038   2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
589039   2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
589040   2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
589041   2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
589042   2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
589043   2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
4675581  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
4675582  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
4675583  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
4675584  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
4675585  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
4675586  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
4675587  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
4675588  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
4675589  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
4675590  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
4675591  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
4675592  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
6710438  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
6710439  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
6710440  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
6710441  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
6710442  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
6710443  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
6710444  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
6710445  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
6710446  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
6710447  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
```

And similarly for client 449747.

```
df[df['MCI_UNIQ_ID'] == 449747]
```

```
         synthetic_data  MCI_UNIQ_ID  CALDR_YR DATE_OF_EVENT  \
5539976  SYNTHETIC DATA       449747      2021    01/31/2021
5539977  SYNTHETIC DATA       449747      2021    03/31/2021
5539978  SYNTHETIC DATA       449747      2021    04/30/2021
5539979  SYNTHETIC DATA       449747      2021    05/31/2021
5539980  SYNTHETIC DATA       449747      2021    06/30/2021
5539981  SYNTHETIC DATA       449747      2021    07/31/2021
5539982  SYNTHETIC DATA       449747      2021    08/31/2021
5539983  SYNTHETIC DATA       449747      2021    09/30/2021
5539984  SYNTHETIC DATA       449747      2021    10/31/2021
5539985  SYNTHETIC DATA       449747      2021    11/30/2021

                                    service  age  GENDER     RACE  \
5539976  Individuals_Receiving_Income_Supports   51  1~Male  1~White
5539977  Individuals_Receiving_Income_Supports   51  1~Male  1~White
5539978  Individuals_Receiving_Income_Supports   51  1~Male  1~White
5539979  Individuals_Receiving_Income_Supports   51  1~Male  1~White
5539980  Individuals_Receiving_Income_Supports   51  1~Male  1~White
5539981  Individuals_Receiving_Income_Supports   51  1~Male  1~White
5539982  Individuals_Receiving_Income_Supports   51  1~Male  1~White
5539983  Individuals_Receiving_Income_Supports   51  1~Male  1~White
5539984  Individuals_Receiving_Income_Supports   51  1~Male  1~White
5539985  Individuals_Receiving_Income_Supports   51  1~Male  1~White

                    ETHNICITY          MARITAL_STATUS EDUCATION_LEVEL
5539976  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
5539977  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
5539978  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
5539979  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
5539980  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
5539981  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
5539982  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
5539983  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
5539984  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
5539985  2~Not Hispanic/Latinx  1~Single-Never Married      99~Unknown
```

4.6.6 Basic Transformations

Often times when we encounter data, we find that it is in a format that is a bit clunky to use. Part of EDA will be to make datasets more friendly. Here we will go over four transformations: substitution, generation, outlier removal, and illogical value removal.

Substitution

For the data entries that have strings in certain columns, like "Individuals_Receiving_DHS_Services", we may need to alter these. Strings can be a bit clunky to work with and are generally read better for plots when we have them in categories. Additionally, we can try and do some mathematical tests on them when we have numerical category labels.

```
#we want to take the columns for gender, race, ethnicity, marital status, and␣
 ↪education level
#and turn these columns into the category data type.

#first we add a new column to the dataframe that has this dtype
#but named with a _cat suffix.
df["GENDER_cat"] = df["GENDER"].astype('category')
df["RACE_cat"] = df["RACE"].astype('category')
df["ETHNICITY_cat"] = df["ETHNICITY"].astype('category')
df["MARITAL_STATUS_cat"] = df["MARITAL_STATUS"].astype('category')
df["EDUCATION_LEVEL_cat"] = df["EDUCATION_LEVEL"].astype('category')
df["service_cat"] = df["service"].astype('category')

df.head()
```

```
   synthetic_data  MCI_UNIQ_ID  CALDR_YR DATE_OF_EVENT  \
0  SYNTHETIC DATA         1626      2021    01/31/2021
1  SYNTHETIC DATA         1626      2021    02/28/2021
2  SYNTHETIC DATA         1626      2021    03/31/2021
3  SYNTHETIC DATA         1626      2021    04/30/2021
4  SYNTHETIC DATA         1626      2021    05/31/2021

                                             service  age  GENDER     RACE  \
0  Children_Attending_Early_Childhood_Programs_Ma...    5  1~Male  1~White
1  Children_Attending_Early_Childhood_Programs_Ma...    5  1~Male  1~White
2  Children_Attending_Early_Childhood_Programs_Ma...    5  1~Male  1~White
3  Children_Attending_Early_Childhood_Programs_Ma...    5  1~Male  1~White
4  Children_Attending_Early_Childhood_Programs_Ma...    5  1~Male  1~White

    ETHNICITY MARITAL_STATUS                EDUCATION_LEVEL GENDER_cat  \
0  99~Unknown     99~Unknown  9-12~High School (grade 9-12)     1~Male
1  99~Unknown     99~Unknown  9-12~High School (grade 9-12)     1~Male
2  99~Unknown     99~Unknown  9-12~High School (grade 9-12)     1~Male
3  99~Unknown     99~Unknown  9-12~High School (grade 9-12)     1~Male
4  99~Unknown     99~Unknown  9-12~High School (grade 9-12)     1~Male

  RACE_cat ETHNICITY_cat MARITAL_STATUS_cat            EDUCATION_LEVEL_cat  \
0  1~White    99~Unknown         99~Unknown  9-12~High School (grade 9-12)
1  1~White    99~Unknown         99~Unknown  9-12~High School (grade 9-12)
2  1~White    99~Unknown         99~Unknown  9-12~High School (grade 9-12)
3  1~White    99~Unknown         99~Unknown  9-12~High School (grade 9-12)
4  1~White    99~Unknown         99~Unknown  9-12~High School (grade 9-12)

                                       service_cat
0  Children_Attending_Early_Childhood_Programs_Ma...
1  Children_Attending_Early_Childhood_Programs_Ma...
2  Children_Attending_Early_Childhood_Programs_Ma...
3  Children_Attending_Early_Childhood_Programs_Ma...
4  Children_Attending_Early_Childhood_Programs_Ma...
```

```
#forgot to add for ids
df["MCI_UNIQ_ID_cat"] = df["MCI_UNIQ_ID"].astype('category')
```

Generation

In other cases, we may want to generate a new variable. These can simply be combinations of previous variables, but in the case of this dataset, we will convert the "age" column to a categorical datatype variable as well. We will do this by putting ages in bins of 10 years.

```
#first we will define a helper function that takes in an int
#and returns the corresponding age bin value (grouped by 10), so a 5 gets returned as
 ↪"0-9"
def age_bin(age):
    if age < 10:
        return "0-9"
    elif age < 20:
        return "10-19"
    elif age < 30:
        return "20-29"
    elif age < 40:
        return "30-39"
    elif age < 50:
        return "40-49"
    elif age < 60:
        return "50-59"
    elif age < 70:
        return "60-69"
    elif age < 80:
        return "70-79"
    elif age < 90:
        return "80-89"
    else:
        return "90+"

#is there a better way to do this? I am not sure, but we will go with this anyway.
#it gets the job done.

#we will now apply this function to the age column and create a new column called AGE_
 ↪BIN
df["AGE_BIN"] = df["age"].apply(age_bin)
df.head()
```

```
   synthetic_data  MCI_UNIQ_ID  CALDR_YR DATE_OF_EVENT  \
0  SYNTHETIC DATA         1626      2021    01/31/2021
1  SYNTHETIC DATA         1626      2021    02/28/2021
2  SYNTHETIC DATA         1626      2021    03/31/2021
3  SYNTHETIC DATA         1626      2021    04/30/2021
4  SYNTHETIC DATA         1626      2021    05/31/2021

                                             service  age  GENDER     RACE  \
0  Children_Attending_Early_Childhood_Programs_Ma...    5  1~Male  1~White
1  Children_Attending_Early_Childhood_Programs_Ma...    5  1~Male  1~White
2  Children_Attending_Early_Childhood_Programs_Ma...    5  1~Male  1~White
3  Children_Attending_Early_Childhood_Programs_Ma...    5  1~Male  1~White
4  Children_Attending_Early_Childhood_Programs_Ma...    5  1~Male  1~White
```

(continues on next page)

(continued from previous page)

```
    ETHNICITY MARITAL_STATUS                EDUCATION_LEVEL GENDER_cat  \
0  99~Unknown     99~Unknown  9-12~High School (grade 9-12)     1~Male
1  99~Unknown     99~Unknown  9-12~High School (grade 9-12)     1~Male
2  99~Unknown     99~Unknown  9-12~High School (grade 9-12)     1~Male
3  99~Unknown     99~Unknown  9-12~High School (grade 9-12)     1~Male
4  99~Unknown     99~Unknown  9-12~High School (grade 9-12)     1~Male

  RACE_cat ETHNICITY_cat MARITAL_STATUS_cat            EDUCATION_LEVEL_cat  \
0  1~White    99~Unknown         99~Unknown  9-12~High School (grade 9-12)
1  1~White    99~Unknown         99~Unknown  9-12~High School (grade 9-12)
2  1~White    99~Unknown         99~Unknown  9-12~High School (grade 9-12)
3  1~White    99~Unknown         99~Unknown  9-12~High School (grade 9-12)
4  1~White    99~Unknown         99~Unknown  9-12~High School (grade 9-12)

                                      service_cat MCI_UNIQ_ID_cat AGE_BIN
0  Children_Attending_Early_Childhood_Programs_Ma...            1626     0-9
1  Children_Attending_Early_Childhood_Programs_Ma...            1626     0-9
2  Children_Attending_Early_Childhood_Programs_Ma...            1626     0-9
3  Children_Attending_Early_Childhood_Programs_Ma...            1626     0-9
4  Children_Attending_Early_Childhood_Programs_Ma...            1626     0-9
```

Another type of generation we can do is to split the date of event so we get the month in which each service occurs. We can do this by simply getting the first two characters of the `DATE_OF_EVENT` column.

```
#make a new month column
#this is the first two characters of the date column and the last 5 characters of the␣
 ↪column
df["month"] = df["DATE_OF_EVENT"].str[:2] + df["DATE_OF_EVENT"].str[-5:]
df.head()
```

```
   synthetic_data  MCI_UNIQ_ID  CALDR_YR DATE_OF_EVENT  \
0  SYNTHETIC DATA         1626      2021    01/31/2021
1  SYNTHETIC DATA         1626      2021    02/28/2021
2  SYNTHETIC DATA         1626      2021    03/31/2021
3  SYNTHETIC DATA         1626      2021    04/30/2021
4  SYNTHETIC DATA         1626      2021    05/31/2021

                                      service  age  GENDER     RACE  \
0  Children_Attending_Early_Childhood_Programs_Ma...    5  1~Male  1~White
1  Children_Attending_Early_Childhood_Programs_Ma...    5  1~Male  1~White
2  Children_Attending_Early_Childhood_Programs_Ma...    5  1~Male  1~White
3  Children_Attending_Early_Childhood_Programs_Ma...    5  1~Male  1~White
4  Children_Attending_Early_Childhood_Programs_Ma...    5  1~Male  1~White

    ETHNICITY MARITAL_STATUS                EDUCATION_LEVEL GENDER_cat  \
0  99~Unknown     99~Unknown  9-12~High School (grade 9-12)     1~Male
1  99~Unknown     99~Unknown  9-12~High School (grade 9-12)     1~Male
2  99~Unknown     99~Unknown  9-12~High School (grade 9-12)     1~Male
3  99~Unknown     99~Unknown  9-12~High School (grade 9-12)     1~Male
4  99~Unknown     99~Unknown  9-12~High School (grade 9-12)     1~Male

  RACE_cat ETHNICITY_cat MARITAL_STATUS_cat            EDUCATION_LEVEL_cat  \
0  1~White    99~Unknown         99~Unknown  9-12~High School (grade 9-12)
1  1~White    99~Unknown         99~Unknown  9-12~High School (grade 9-12)
2  1~White    99~Unknown         99~Unknown  9-12~High School (grade 9-12)
```

(continues on next page)

(continued from previous page)

```
3  1~White    99~Unknown         99~Unknown  9-12~High School (grade 9-12)
4  1~White    99~Unknown         99~Unknown  9-12~High School (grade 9-12)

                                        service_cat MCI_UNIQ_ID_cat AGE_BIN  \
0  Children_Attending_Early_Childhood_Programs_Ma...            1626     0-9
1  Children_Attending_Early_Childhood_Programs_Ma...            1626     0-9
2  Children_Attending_Early_Childhood_Programs_Ma...            1626     0-9
3  Children_Attending_Early_Childhood_Programs_Ma...            1626     0-9
4  Children_Attending_Early_Childhood_Programs_Ma...            1626     0-9

     month
0  01/2021
1  02/2021
2  03/2021
3  04/2021
4  05/2021
```

Outlier Removal

Often times, data will have outliers. These are data points that are very far from the majority of the data. In the case of a dataset with all categorical variables, this is not often feasible. However, such an act is more common with numeric features. Let's try the age category and remove all the rows with age 105 or greater.

```
#want to remove all entries from the dataset where the age entry is 105 or more
df = df[df["age"] < 105]
df.shape
```

```
(7115529, 20)
```

Illogical Value Removal

This portion of data transformations is slightly harder as we will have to look at multiple features at the same time. However, it is still important. Some data points in the dataset have children of age 5, yet lists them as in high school.

```
#we want to remove all the values where the age is 0-9
#AND where they are listed as in high school as "9-12~High School (grade 9-12)"
df = df[~((df["AGE_BIN"] == "0-9") & (df["EDUCATION_LEVEL"] == "9-12~High School␣
↪(grade 9-12)"))]
df.shape
```

```
(6581506, 20)
```

4.6.7 Advanced Transformations

Sometimes when doing exploratory data analysis, the storage format of the table does not give a complete view of the picture. In the case of this dataset, we found earlier that there were several rows with the same ID, which means that there are multiple entries for some people. This indicates that they used the services provided in the dataset multiple times.

To get a better picture of some data insights, we can make a new dataframe; this one will have one row for each ID, and we will add the columns regarding the number of services used, number of distinct services used, and the number of months the services were used. Because the dataset is big, this may take a while.

In general cases, to find new transformations for the datasets you work with, you will need to think about the general direction to take the data, which can involve simply thinking about the data logically. However, a better approach is to consult domain knowledge. This basically involves looking to sources related to the data (research articles, people who collected the data, people who are knowledgeable on the subject) and transforming the data as they may advise, or getting advice on where to look for ideas on transformations.

Here, we are going to perform two aggregative transformations. These will group the entries of the dataset by the ID, as well as by the (ID, Month) pairing and compute some summary statistics, including the number of services used, the number of months the services were used, and the range of dates the services were used.

```
def get_client_attributes(services):
    client_attributes = services.groupby(['MCI_UNIQ_ID']).agg(
        num_services = ('service', 'count'),
        distinct_services = ('service', 'nunique'),
        first_date = ('DATE_OF_EVENT', 'min'),
        last_date = ('DATE_OF_EVENT', 'max'),
        num_months = ('month', 'count'),
        distinct_month = ('month', 'nunique')
    ).reset_index()

    return client_attributes

client_attributes = get_client_attributes(df)
client_attributes.head()
```

	MCI_UNIQ_ID	num_services	distinct_services	first_date	last_date	num_months	distinct_month
0	1	12	1	01/31/2021	12/31/2021	12	12
1	2	1	1	02/28/2021	02/28/2021	1	1
2	3	11	1	02/28/2021	12/31/2021	11	11
3	5	12	1	01/31/2021	12/31/2021	12	12
4	6	24	3	01/31/2021	12/31/2021	24	12

Similarly, we perform the client-month attribute aggregations.

```
def get_client_date_attributes(services):
    client_date_attributes = services.groupby(['MCI_UNIQ_ID', 'month']).agg(
        num_services = ('service', 'count'),
        distinct_services = ('service', 'nunique')
    ).reset_index()

    return client_date_attributes
```

(continues on next page)

(continued from previous page)

```
client_date_attributes = get_client_date_attributes(df)
client_date_attributes.head()
```

```
   MCI_UNIQ_ID    month  num_services  distinct_services
0            1  01/2021             1                  1
1            1  02/2021             1                  1
2            1  03/2021             1                  1
3            1  04/2021             1                  1
4            1  05/2021             1                  1
```

Now, we can merge the datasets in a sense so that some of the information about each person can carry over to the new datasets we've created. Specifically, we want to carry over gender, race, ethnicity, education level, and marital status.

```
#start with client_attributes
#we want to add columns of age, gender, race, ethnicity, education level, and marital␣
 ↪status
#from the original dataset, making sure IDs match up and that there is only one␣
 ↪occurrence of each ID.
client_attributes = pd.merge(client_attributes, df[['MCI_UNIQ_ID', 'age', 'GENDER',
 ↪'RACE', 'ETHNICITY', 'MARITAL_STATUS', 'EDUCATION_LEVEL']], on='MCI_UNIQ_ID', how=
 ↪'left')
#now remove duplicate rows
client_attributes = client_attributes.drop_duplicates(subset=['MCI_UNIQ_ID'])
client_attributes.head()
```

```
    MCI_UNIQ_ID  num_services  distinct_services  first_date   last_date  \
0             1            12                  1  01/31/2021  12/31/2021
12            2             1                  1  02/28/2021  02/28/2021
13            3            11                  1  02/28/2021  12/31/2021
24            5            12                  1  01/31/2021  12/31/2021
36            6            24                  3  01/31/2021  12/31/2021

    num_months  distinct_month  age    GENDER                     RACE  \
0           12              12   13    1~Male                99~Unknown
12           1               1   70  2~Female                   1~White
13          11              11   29  2~Female                   1~White
24          12              12   40    1~Male  2~Black/African American
36          24              12   71  2~Female  2~Black/African American

                ETHNICITY          MARITAL_STATUS  \
0   2~Not Hispanic/Latinx              99~Unknown
12  2~Not Hispanic/Latinx               2~Married
13  2~Not Hispanic/Latinx               2~Married
24  2~Not Hispanic/Latinx               2~Married
36  2~Not Hispanic/Latinx  1~Single-Never Married

                          EDUCATION_LEVEL
0   GED~High School Diploma/GED Completed
12                             99~Unknown
13          9-12~High School (grade 9-12)
24                 UNDERGRAD~Some College
36                             99~Unknown
```

```
#we'll do the same with client_date_attributes
```

(continues on next page)

(continued from previous page)

```
client_date_attributes = pd.merge(client_date_attributes, df[['MCI_UNIQ_ID', 'age',
 ↪'GENDER', 'RACE', 'ETHNICITY', 'MARITAL_STATUS', 'EDUCATION_LEVEL']], on='MCI_UNIQ_
 ↪ID', how='left')
#now remove rows that have duplicate ID-month pairs
client_date_attributes = client_date_attributes.drop_duplicates(subset=['MCI_UNIQ_ID',
 ↪ 'month'])
client_date_attributes.head()
```

```
    MCI_UNIQ_ID    month  num_services  distinct_services  age  GENDER  \
0             1  01/2021             1                  1   13  1~Male
12            1  02/2021             1                  1   13  1~Male
24            1  03/2021             1                  1   13  1~Male
36            1  04/2021             1                  1   13  1~Male
48            1  05/2021             1                  1   13  1~Male

          RACE              ETHNICITY MARITAL_STATUS  \
0   99~Unknown  2~Not Hispanic/Latinx     99~Unknown
12  99~Unknown  2~Not Hispanic/Latinx     99~Unknown
24  99~Unknown  2~Not Hispanic/Latinx     99~Unknown
36  99~Unknown  2~Not Hispanic/Latinx     99~Unknown
48  99~Unknown  2~Not Hispanic/Latinx     99~Unknown

                          EDUCATION_LEVEL
0   GED~High School Diploma/GED Completed
12  GED~High School Diploma/GED Completed
24  GED~High School Diploma/GED Completed
36  GED~High School Diploma/GED Completed
48  GED~High School Diploma/GED Completed
```

4.6.8 Saving Data Transformations

If you want to close your work and come back to it easily without having to rerun all of the above code, or if you want to use the same data in a new notebook, an easy workaround is to save the data to an intermediate file and then reload it. We'll be saving all of our datasets to files that we will then plot in the next section.

```
#we will save df to a new csv file
df.to_csv('dhs_cleaned.csv', index=False)
client_attributes.to_csv('dhs_client_attributes.csv', index=False)
client_date_attributes.to_csv('dhs_client_date_attributes.csv', index=False)
```

4.6.9 EDA Project: Part 1

Answer the following questions:

- Which MCI_UNIQ_ID has the highest number of records among all IDs?
- What is the average number of interactions for clients in the file?
- How many months did a particular client use the service?
- What other general observations can you make from the data?

4.7 Plotting and Statistical Testing II

4.7.1 Plotting

Making visualizations is a key part of EDAs. So in order to plot, we will use packages `matplotlib` and `seaborn` to make some nice-looking plots.

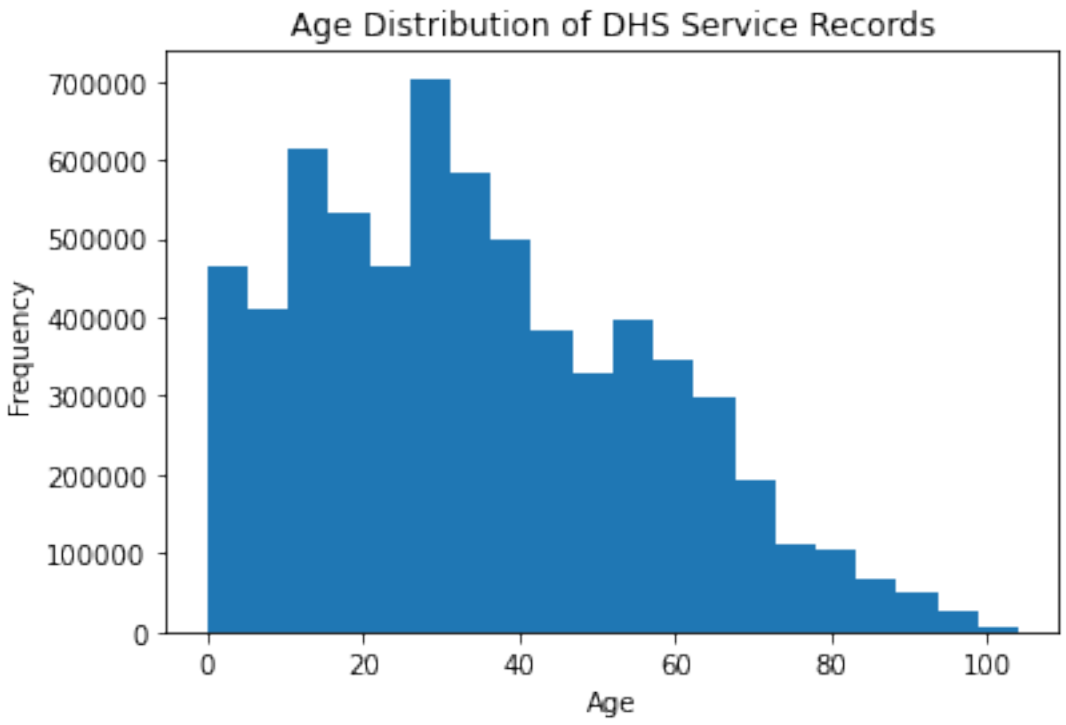
The type of plot made typically depends on the type of the data used, whether it is a discrete category or a continuous number, as well as the number of variables analysed in a single plot. We will go over some good plots for categorical data later in this section, but a reference table for common plots is given below.

	One Variable	Two Variables
Discrete	Bar Plot	Countplot, Jitterplot, Tornado/Butterfly Plot
Continuous	Histogram, Density, Dotplot	Scatterplot, Fitted Curve, Density Plot
One of Each		Column Plot, Boxplot, Violin Plot

Histograms

Histograms are a good way of looking at the overall distribution of a single feature, or column.

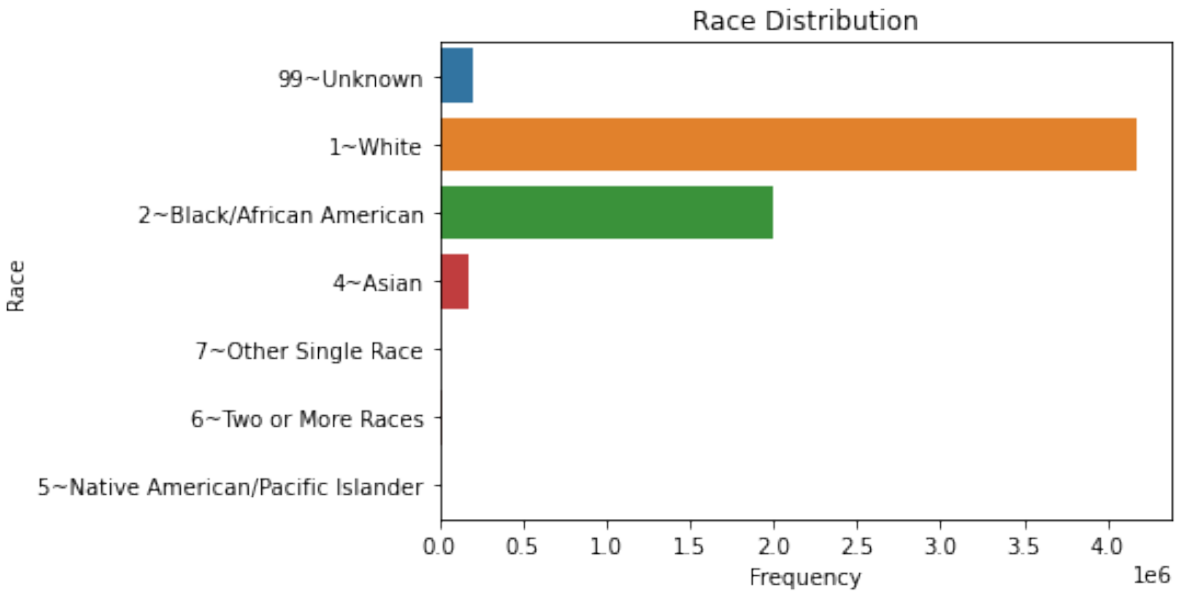
```
#here we want to make a histogram of the age column in the dataset
#using matplotlib.pyplot
import matplotlib.pyplot as plt

#since we already removed NA values, we can get straight to plotting
plt.hist(df['age'], bins=20)
#add some labels
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.title('Age Distribution of DHS Service Records')
plt.show()
```



Barplots

To look at the distribution between distinct categories of a discrete variable, we can turn to bar plots. These can be vertical or horizontal, but with many categories, we will want to make the plot horizontal for readability.

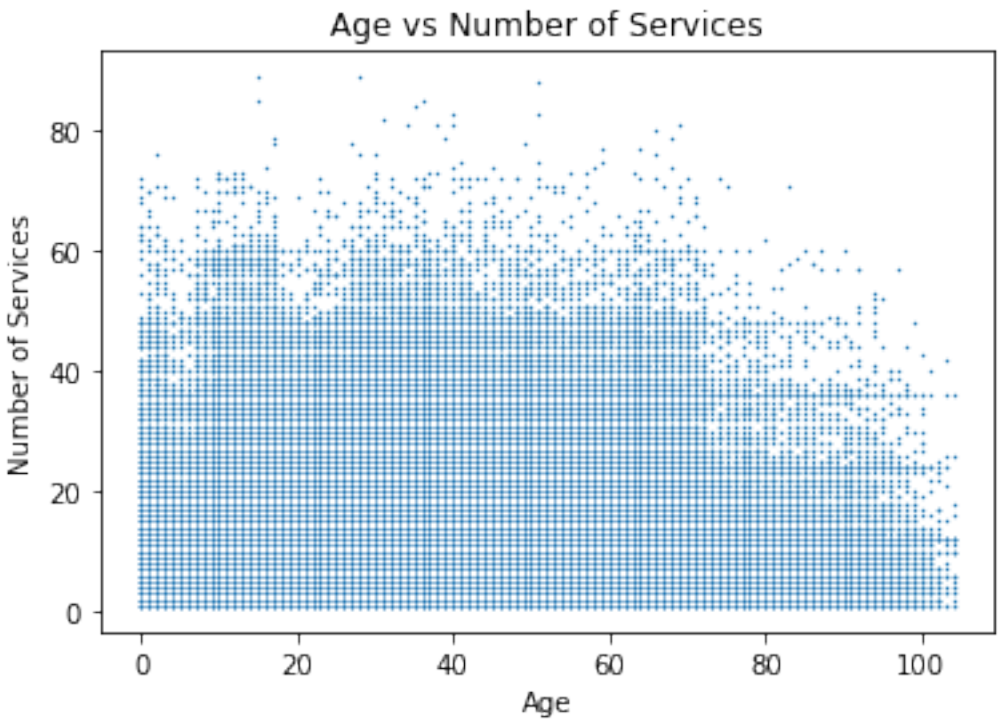
```
import seaborn as sns

sns.countplot(y = df['RACE'])
plt.title('Race Distribution')
plt.ylabel('Race')
plt.xlabel('Frequency')
plt.show()
```



Scatter Plots

For looking at the relationship between two numerical variables, we turn to scatterplots. We can also add trend lines if we think there may be a relationship between the two.

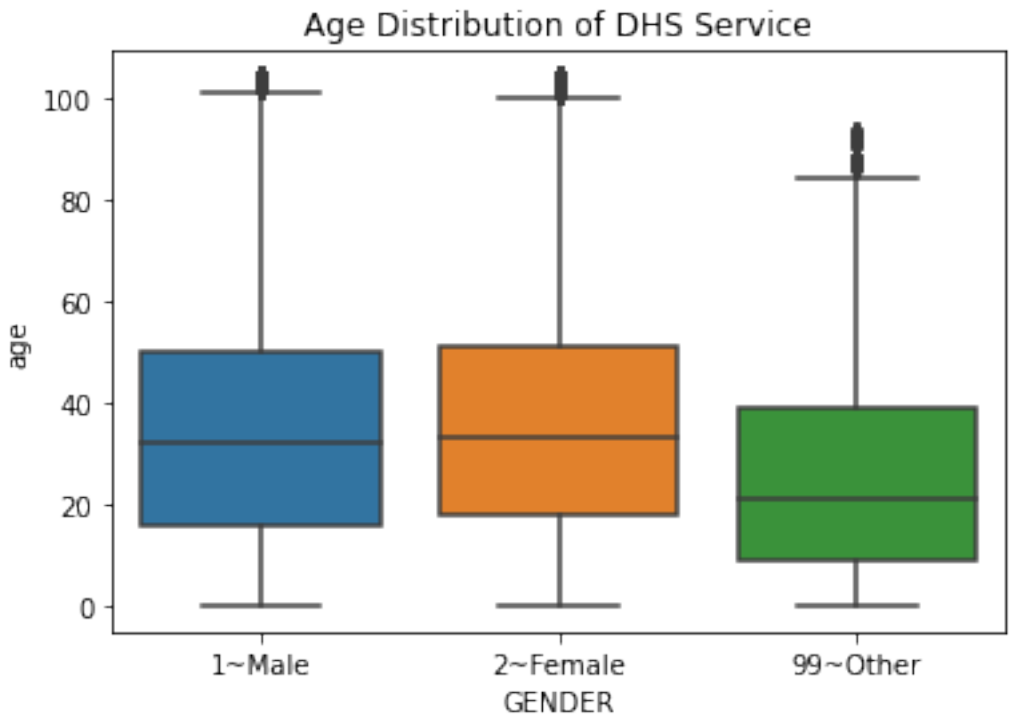
```
#make a scatterplot between age and number of services in the client_attributes␣
↪dataset

sns.scatterplot(data = client_attributes, x = 'age', y = 'num_services', s = 2)
plt.title('Age vs Number of Services')
plt.xlabel('Age')
plt.ylabel('Number of Services')
plt.show()
```



Boxplots

For looking at the relationship between two variables, one numeric and one categorical, we can use boxplots, which also give some information on the quartiles of the numeric variable.

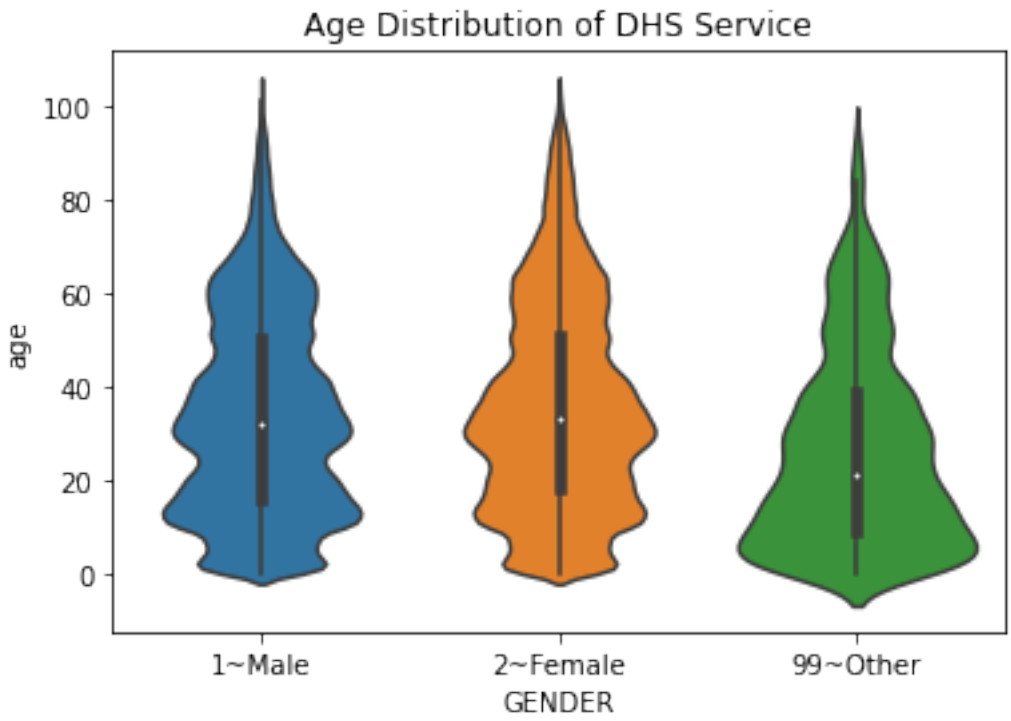
```
#we want to make a boxplot of age, separated by gender
#we can use seaborn for this
import seaborn as sns

sns.boxplot(data = df, x = "GENDER", y = "age")
plt.title('Age Distribution of DHS Service')
plt.show()
```



Violin plots

A variation on the boxplot is the violin plot, which shows the relative distributions of each of the groups you are examining.

```
#we want to make the same as above, but with violin plots
sns.violinplot(data = df, x = "GENDER", y = "age")
plt.title('Age Distribution of DHS Service')
plt.show()
```

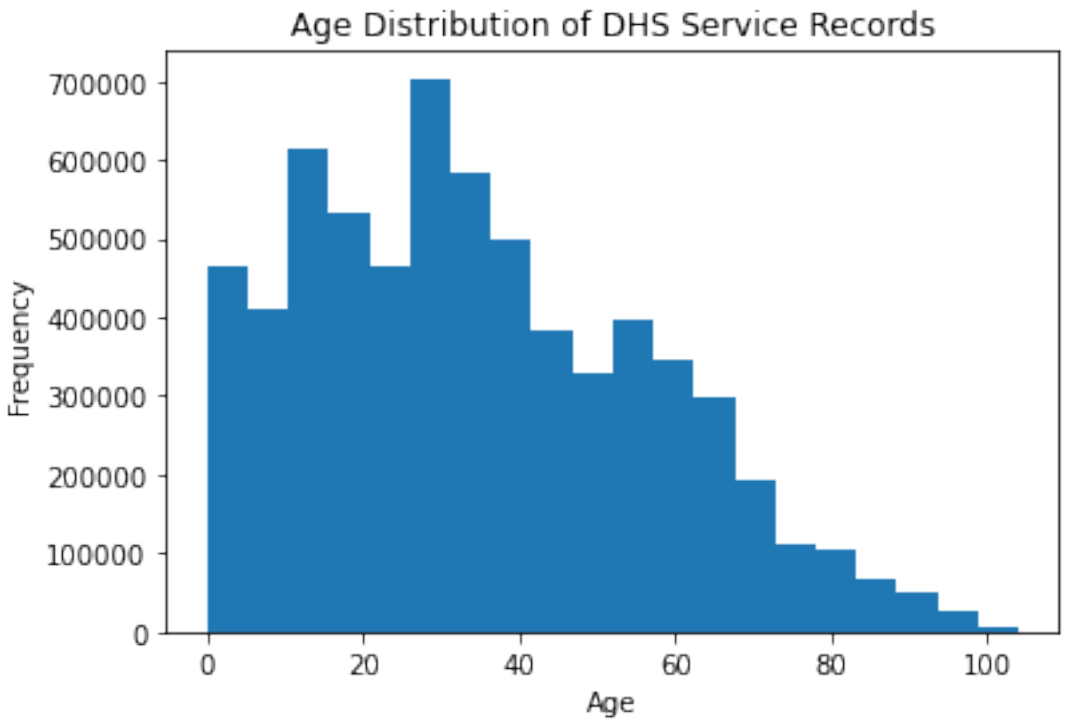


4.7.2 Interpreting Plots

Plotting is one thing, but plots alone do not make for good EDAs. Plots must be interpreted as well. This part can be subjective, but with each iteration of EDA, you will hopefully get better and better interpretations.

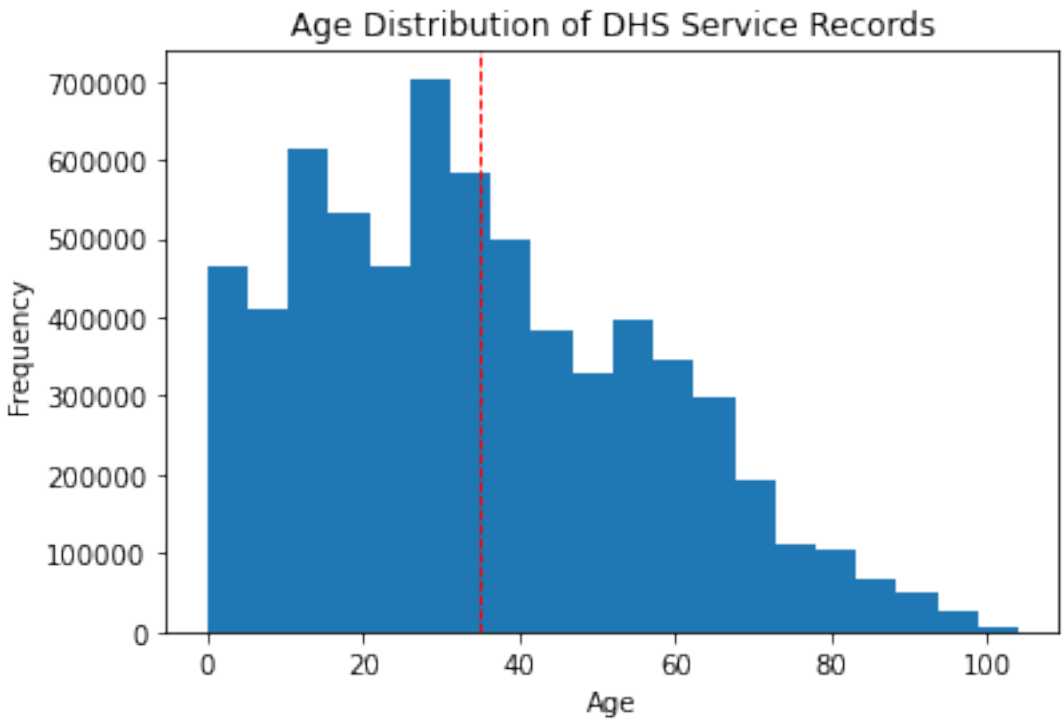
If we refer to the histogram of ages for this dataset, what can we interpret?

```
plt.hist(df['age'], bins=20)
#add some labels
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.title('Age Distribution of DHS Service Records')
plt.show()
```



We can see that the distribution of ages is right-skewed but is somewhat normally distributed. The mean and mode seem to be around the late 30's. But we can be more objective with our interpretations by adding these statistics to the plot.

```
#remake previous histogram
#but add a line for the mean of the distribution
plt.hist(df['age'], bins=20)
#add some labels
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.title('Age Distribution of DHS Service Records')
plt.axvline(df['age'].mean(), color='red', linestyle='dashed', linewidth=1)
plt.show()
```



Exercise 6.1

Repeat the above, adding the plot's estimated normal distribution.

The exercise above asks to evaluate, visually, if this distribution approximately follows a normal distribution. The normal distribution is important for a myriad of statistical tests, which are discussed in the next section.

4.7.3 Statistical Testing

While visualizations make for good reports and are a good way of visually looking at data, you can often get much more conclusive results about data by doing statistical tests. All statistical tests follow a principle of deviation from a *null hypothesis* H_0; this null hypothesis is what the tester assumes to be true. The tester then calculates the probability of the observations being the way they are given the null hypothesis; if the probability calculated is low, the tester can reject the null hypothesis.

Warning

This is a massive simplification of the theory behind statistical testing. For more accurate theory, refer to the section on Statistical Inference.

There are a myriad of statistical tests, most of which apply to numerical data, but a few of them apply to categorical data. As with plotting, different ones apply in different situations: whether or not we are comparing one or two samples, whether these samples follow a normal distribution, etc.

Numerical Comparisons

In general, numerical comparison tests follow the null hypothesis of:

- (for one variable) The mean of the dataset is equal to a specific value
- (for two or more variables) The difference in the means of the groups is equal to zero

Tests involving numerical comparisons are split on how many groups are compared and whether or not these groups follow a normal distribution. The normal distribution assumption is critical for some of these tests; this is because the normal distribution was assumed in the making of these tests and therefore may give you incorrect conclusions if the data do not follow normal distributions when you use them.

	Analyzing One Variable	Analyzing Two Variables	Analyzing 3+ Variables
Normality Assumed	One-sample t-test	Two-sample t-test	ANOVA
Normality Not Assumed	Sign test	Mann-Whitney U-test, Wilcoxon signed-rank test	Kruskal-Wallis test

One Group Comparisons

To compare one variable to a set null hypothesis, we use a one-sample t-test (if the data follow a normal distribution) or a sign test (if not). Here, let's look at the number of services used by each person in the dataset, and we'll set the null hypothesis that these means are equal to 6.

```
#for the num_services column in the client_attributes dataset
#perform one-sample t-test and sign test
from scipy.stats import ttest_1samp
from scipy.stats import wilcoxon

#perform one-sample t-test
t_stat, p_val = ttest_1samp(client_attributes['num_services'], 6)
print('One-sample t-test results:')
print('t-statistic:', t_stat)
print('p-value:', p_val)
print("")

#perform sign test
t_stat, p_val = wilcoxon(client_attributes['num_services'] - 6)
print('Sign test results:')
print('Test statistic:', t_stat)
print('p-value:', p_val)
```

```
One-sample t-test results:
t-statistic: 628.1046487314816
p-value: 0.0

Sign test results:
Test statistic: 2322208324.0
p-value: 0.0
```

These statistics are indicative of the mean number of services used being different from 6. We would need to perform additional visualizations or testing to get more insights than this, however.

Two Group Comparisons

To compare two groups of data, we would use a paired t-test or a Mann-Whitney U-Test depending on if the groups are normally distributed or not, respectively. The null hypothesis here is that the difference in means between the two groups is some value, typically zero.

Here, let's split the number of services used by gender. Note that this dataset has some entries where the gender is listed as "other", which we saw in plotting. For instructional purposes, we will remove these entries; in practice, we will want to include these with ANOVA or Kruskal-Wallis, which is discussed in the next section.

```
#t-test and mann-whitney u test for num-services split by gender
#removing "other" entries in gender
intermediate_client_atts = client_attributes[client_attributes['GENDER'] != '99~Other
 ↪']

#perform t-test
from scipy.stats import ttest_ind
m = intermediate_client_atts[intermediate_client_atts['GENDER'] == '1~Male']['num_
 ↪services']
f = intermediate_client_atts[intermediate_client_atts['GENDER'] == '2~Female']['num_
 ↪services']

t_stat, p_val = ttest_ind(m, f)
print('T-test results:')
print('t-statistic:', t_stat)
print('p-value:', p_val)
print("")

#perform mann-whitney u test
from scipy.stats import mannwhitneyu
t_stat, p_val = mannwhitneyu(m, f)
print('Mann-Whitney U test results:')
print('Test statistic:', t_stat)
print('p-value:', p_val)
```

```
T-test results:
t-statistic: -1.205624178437287
p-value: 0.22796301465831825

Mann-Whitney U test results:
Test statistic: 29664099747.5
p-value: 1.5309720336376268e-06
```

We can see here that these tests give very different p-values. It is best practice then to see which test is more accurate, and we would do this by testing the normality of each group. We can do this with a normality test (discussed later in this section) or by using histograms for visual confirmation.

Exercise 6.2

For the groups tested above, make the histograms for each and judge which test yields a more accurate conclusion.

Many-Group Comparisons

To compare three or more groups, we could do t-tests or Mann-Whitney U-tests between each pairing of groups. However, this yields a higher possibility of what are known as Type I Errors (rejecting a null hypothesis when we should not reject it). The mathematical theory as to why this is the case is not important; what is more important is that we have ways to circumvent this.

Again, for normally and non-normally distributed data, we use Analysis of Variance (ANOVA) or a Kruskal-Wallis Test, respectively. Here, let's look at the number of services used split by race.

```
#perform anova and kruskal-wallis test for num-services split by race
#not removing any values this time
import statsmodels.api as sm
from statsmodels.formula.api import ols

#perform anova
model = ols('num_services ~ RACE', data = client_attributes).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
print('ANOVA results:')
print(anova_table)
print("")

#perform kruskal-wallis test
from scipy import stats
r_unknown = client_attributes[client_attributes['RACE'] == '99~Unknown']['num_services
 ↪']
r_white = client_attributes[client_attributes['RACE'] == '1~White']['num_services']
r_black = client_attributes[client_attributes['RACE'] == '2~Black/African American'][
 ↪'num_services']
r_asian = client_attributes[client_attributes['RACE'] == '4~Asian']['num_services']
r_native = client_attributes[client_attributes['RACE'] == '5~Native American/Pacific␣
 ↪Islander']['num_services']
r_more = client_attributes[client_attributes['RACE'] == '6~Two or More Races']['num_
 ↪services']
r_other = client_attributes[client_attributes['RACE'] == '7~Other Single Race']['num_
 ↪services']

t_stat, p_val = stats.kruskal(r_unknown, r_white, r_black, r_asian, r_native, r_more,␣
 ↪r_other)
print('Kruskal-Wallis test results:')
print('Test statistic:', t_stat)
print('p-value:', p_val)
```

```
ANOVA results:
                sum_sq        df           F         PR(>F)
RACE      9.701921e+04       6.0  236.969478  1.271657e-303
Residual  3.346570e+07  490440.0         NaN            NaN

Kruskal-Wallis test results:
Test statistic: 1219.5315398715004
p-value: 2.836686948222061e-260
```

Both of these tests have low p-values, indicating that there is a significant difference within at least one pairing. To determine where this difference is, we have to use what are called post-hoc tests. For ANOVA, we use TukeyHSD Post-hoc tests, and for Kruskal-Wallis, we use Dunn tests. Here, we only look at TukeyHSD, especially as post-hoc tests for Kruskal-Wallis are less common, which is understandable given that Kruskal-Wallis tests are less common as a whole.

```
#post-hoc tests for the anova test
from statsmodels.stats.multicomp import pairwise_tukeyhsd

tukey = pairwise_tukeyhsd(endog = client_attributes['num_services'], groups = client_
↪attributes['RACE'], alpha = 0.05)
print(tukey.summary())
```

```
                            Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================================================
              group1                              group2               meandiff p-
↪adj   lower   upper  reject
-------------------------------------------------------------------------------------
↪-------------------------
                           1~White           2~Black/African American   0.4477  0.
↪001  0.3702  0.5251   True
                           1~White                            4~Asian  -1.4977  0.
↪001 -1.7021 -1.2932   True
                           1~White 5~Native American/Pacific Islander  -1.4682 0.
↪0625 -2.9764  0.0399  False
                           1~White                6~Two or More Races  -0.2988   ␣
↪0.9 -1.0409  0.4432  False
                           1~White                7~Other Single Race  -1.3499 0.
↪0108 -2.5105 -0.1894   True
                           1~White                         99~Unknown  -1.4201  0.
↪001 -1.6131  -1.227   True
          2~Black/African American                            4~Asian  -1.9453  0.
↪001 -2.1551 -1.7356   True
          2~Black/African American 5~Native American/Pacific Islander  -1.9159 0.
↪0034 -3.4248  -0.407   True
          2~Black/African American                6~Two or More Races  -0.7465 0.
↪0483   -1.49  -0.003   True
          2~Black/African American                7~Other Single Race  -1.7976  0.
↪001 -2.9591 -0.6361   True
          2~Black/African American                         99~Unknown  -1.8677  0.
↪001 -2.0664 -1.6691   True
                           4~Asian 5~Native American/Pacific Islander   0.0294   ␣
↪0.9 -1.4912  1.5501  False
                           4~Asian                6~Two or More Races   1.1988  0.
↪001  0.4316   1.966   True
                           4~Asian                7~Other Single Race   0.1477   ␣
↪0.9 -1.0291  1.3246  False
                           4~Asian                         99~Unknown   0.0776   ␣
↪0.9 -0.1967  0.3519  False
5~Native American/Pacific Islander                6~Two or More Races   1.1694 0.
↪3829 -0.5103  2.8491  False
5~Native American/Pacific Islander                7~Other Single Race   0.1183   ␣
↪0.9 -1.7837  2.0203  False
5~Native American/Pacific Islander                         99~Unknown   0.0482   ␣
↪0.9  -1.471  1.5673  False
               6~Two or More Races                7~Other Single Race  -1.0511 0.
↪2678 -2.4272   0.325  False
               6~Two or More Races                         99~Unknown  -1.1212  0.
↪001 -1.8855  -0.357   True
               7~Other Single Race                         99~Unknown  -0.0701   ␣
↪0.9  -1.245  1.1047  False
-------------------------------------------------------------------------------------
↪-------------------------
```

Categorical Analysis: Chi-Square Tests of Independence

A good way to compare two categorical datatypes is to use a chi-square test of independence. This can easily be called from packages in `scipy.stats`.

This test operates on the null hypothesis that the two categories are independent of each other; in other words, there exists no relationship between the two chosen variables. It creates a contingency table, splitting each variable on its own axis and putting the number of occurrences of each kind of data point in its respective box. For example:

	Apples	Bananas
Fresh	30	27
Frozen	5	3

The above contingency table indicates that there are 30 occurrences of fresh apples in our pretend dataset. This data may have come from a dataset such as

	Fruit Type	Quality
0	Apple	Fresh
1	Banana	Fresh
2	Apple	Fresh
3	Apple	Frozen
…	…	…
64	Banana	Frozen

The contingency table then adds subtotals to each row and column; for example:

	Apples	Bananas	Total
Fresh	30	27	57
Frozen	5	3	8
Total	35	30	65

It then calculates an expected value for each entry by multiplying that entry's row and column totals, then dividing it by the total number of occurrences. So the expected number of fresh apples would be $\frac{35*57}{65} = 30.69$. This is done for all entries; the two tables are then compared for deviation from each other.

For a dataset such as what we are using, doing this would be impossible by hand, but computers make it easy.

```
#we are going to do a chi-square test of independence
#to see if there is a relationship between RACE and EDUCATION_LEVEL
#we can use the scipy library for this
from scipy.stats import chi2_contingency

#first we get the categories from the race column
race_cats = df["RACE_cat"].astype('category').cat.categories
race = pd.Categorical(df["RACE_cat"], categories=race_cats)
#then same for education
edu_cats = df["EDUCATION_LEVEL_cat"].astype('category').cat.categories
edu = pd.Categorical(df["EDUCATION_LEVEL_cat"], categories=edu_cats)
```

(continues on next page)

(continued from previous page)

```
#now the contingency table
tab = pd.crosstab(race, edu)

#now we can run the chi-square test
chi2, p, dof, expected = chi2_contingency(tab)
print("Chi-square test of independence: p = ", p, "chi2 = ", chi2)
```

```
Chi-square test of independence: p =  0.0 chi2 =  19200.2208703978
```

This incredibly high statistic and this very low p-value indicates that there is a dependency between race and education level.

While other categorical data tests exist, such as Fisher's Exact test and McNemar's test, they are not used very often in practice, especially for EDA.

Normality Tests

In practice, tests comparing numerical data may give you the same result regardless of what test you use. However, to assume that this will be the case is not good practice. It is better to perform both tests and analyze differences; if differences between the two tests' conclusions arise, then it is good to perform normality tests to narrow down which conclusion is more accurate.

Normality tests operate on the null hypothesis that there is no deviation from a normal distribution for a given dataset. Its associated p-value is low if there is significant deviation from a normal distribution and high if the data follow an approximate normal distribution.

There are many tests to analyze normality, but the one with the most power is the Shapiro-Wilk test. We'll use this test to look at the normality of the number of services used overall.

```
#shapiro-wilk test for normality of num_services in client_attributes
from scipy.stats import shapiro

stat, p = shapiro(client_attributes['num_services'])
print('Shapiro-Wilk test results:')
print('Test statistic:', stat)
print('p-value:', p)
```

```
Shapiro-Wilk test results:
Test statistic: 0.6876380443572998
p-value: 0.0
```

```
/opt/anaconda3/lib/python3.9/site-packages/scipy/stats/morestats.py:1760:␣
↪UserWarning: p-value may not be accurate for N > 5000.
  warnings.warn("p-value may not be accurate for N > 5000.")
```

This p-value indicates that it is likely this distribution is not normal. To confirm this is the case, however, we should also make a histogram.

4.7.4 EDA Project, Part 2

EDA typically involves many more tools and statistical techniques than are described in this chapter. As such, refer to outside documentation, such as for `scipy.stats`, and look into the following:

- Convert all date variables into the date datatype from their current string type.
- Summarize the date/ID paired client attribute dataset. How many clients used the services in January? How many clients used services by each month?
- What is the average number of services a client use?
- How many clients receive services every month?
- Does the number of services per client increase over time?
- Do the statistics on the columns make sense? For example, are there any inconsistencies between age and education level?

Secondly, perform many plots and tests and isolate the ones you believe to carry the most notable results.

4.8 Making EDA Reports

When you are doing EDA, it is often the case (whether in academia or industry) that you will have to put your findings into a report rather than just sending a Jupyter Notebook with code, plots, and test results. This report should also read nicely; well-written reports are more likely to make an impact on readers.

In this section, we will go over basic writing principles for EDA and preparing your Jupyter Notebooks for creating a report output.

4.8.1 EDA Writing Principles

Many people are familiar with the Scientific Method, which formulates a cyclical workflow for performing science. This process involves:

1. Making preliminary observations and reading prior research;
2. Formulating hypotheses based on these observations and research;
3. Designing and performing experiments to test these hypotheses;
4. Evaluating the results collected from the experiments;
5. Making conclusions based on the results, which cycles back into the first step.

In scientific research papers, these phases formulate into sections for the report. Initial observations, prior research, and hypotheses form an **Introduction** section. The experiments performed form a **Methods** section, and its results form a **Results** section. Following this, the conclusions reached from the results form a **Discussion** section.

Data science, being an offshoot of science, benefits from this workflow as well. However, EDA reports follow a modified structure, placing less emphasis on the methods and requiring less rigor for introduction and discussion sections. Because of this, EDA reports are not typically published but rather can form a piece of a larger work.

Here, we will go over some of the sections of EDA in-depth, and how to write them.

Introduction

The introduction of the EDA should answer, at the very least, the following questions:

- Where did you get the data from?
- What are your guiding questions throughout this process?
- Why did you form these questions?

The introduction section can also include, especially for reports that are made more visible to the public, some background on the topic in general so as to help readers without prior knowledge of the subject get more understanding as to what the topic is about and why it is important. This means incorporating preliminary observations and looking into prior research if possible.

Data Preparation

You undoubtedly had to perform some previous data manipulation to get to the basic analysis portion of your EDA. While you do not need to include any code in your EDA report or include basic preprocessing like removing N/A values, you should highlight any methods you took that are atypical, such as transformation to make new variables. The rationale for doing this is also important in this section.

In other words, this section should answer:

- What did you do to the data that is notable?
- Why did you do it?

Results

Rather than have a methods section where you introduce the tests and plots that you made, it is better to segment these aspects together as you go through the results. One framework you can take through this is to go through each of your results, or alternatively, each of the features you found to have striking results. This framework would lend itself to a sub-structure as follows:

- Feature title
  - Feature description
  - Important result
  - Plots and tests to back up your result
  - Interpretations of the result
- Next feature title…

This is likely to be the most important section of your report, so put as much effort into it as possible. Make sure it is polished and that your rationale and results make sense.

This section should answer, in short:

- What did you find?
- What do your findings mean?

Discussion

This section does not have to be labeled as a "Discussion" section, and it will likely not be as extensive as it typically is in many formal research papers. However, it should tackle the questions:

- What did you find, overall? (Summarize your results!)
- Why are your findings important?
- What next steps could be taken in analysis?

Such next steps could be ones that come to you as you are performing analysis, but often times you can get these questions as you get feedback from others on your analysis. Feedback is important for the overall strength of a report; an additional set of eyes can see errors that you may not have caught when writing the report or even in revision (more likely than you think - with your own writing, you may be tempted to skip over a section especially if you think you know that section well).

4.8.2 General Writing Tips

With many reports, one big tip is to elaborate as much as possible. Adding more content is generally better than adding less content; this is because in writing (like in many aspects of life) it is easier to take things away than to add something completely new.

Another tip is to not take feedback poorly. As mentioned previously, feedback is important and will help you to improve your analysis with each iteration. Though sometimes feedback can be quite harsh, this often points out areas for significant improvement.

Writing for EDA reports and for scientific reports in general is very technical. There is typically no need to try and make a report pleasing to read; rather, it is more important that the report is not confusing and gets all the details that are important to cover. That being said, being able to succinctly state what you need to say is good, as extra unnecessary text can make your writing more confusing.

Finally, I encourage you to enjoy the process. With many things in life, you will get more satisfaction out of the process, *and* you will do a better job, if you enjoy writing reports. For some people, writing is not their strength, or something they particularly enjoy. Fortunately, there are many people out there who are willing and able to help you out. Having additional people look at your reports or your progress over time can open your eyes to new insights about your own work that you may have missed, such as your own improvement with writing.

4.8.3 Notebook Preparation

Thankfully for EDA report-writers, Jupyter Notebooks have the amazing capability of being able to utilize markdown in addition to running code. This framework of markdown cells and code cells forms the foundation of report-writing, upon which additional cleanup is necessary to make a presentable report.

Here, we will go over some markdown basics, hiding code, and outputting a PDF from your notebook.

Markdown Basics

Here, we introduce some of the more common features of markdown that may be helpful to you as you make reports.

One of the biggest features of markdown to use is the making of titles. This will help you with sectioning your report and will thus make it easier to read. Typically, reports should have one main title at the top, followed by section titles and sub-section titles. These are all indicated with the hash symbol, pound sign, or hashtag, whatever you want to call it (#). One symbol is used for the main title, two for sections, and three for sub-sections. You can use even more for sub-subsections if you need.

Some people like to write some words in italics or boldface to emphasize points. This is done with asterisks surrounding the portion you want to italicize or make bold, with one asterisk making italics and two making boldface. For example,

- *This text will be italicized.*
- **This text will be in bold.**

Lists are another common tool in making reports and are done with the hyphen character (-) or a number followed by a period (like "1.") at the beginning of a new line.

Markdown also has cross-functionality with LaTeX and HTML. Either of these can be used within markdown in most cases. LaTeX is good for implementing mathematical formulas; in-line math is done with $ surrounding a LaTeX math formula, while large formulas can be done on their own lines by a $$ on a new line, followed by a LaTeX formula on the next line, and another $$ on the next line. HTML tags typically have synonymous functions in either LaTeX or markdown; due to the clunky nature of HTML tags, they are not recommended to be used unless you know HTML very well.

Finally, some characters like the asterisk and the dollar sign are special, as you saw in the prior paragraphs. To print these characters as they are and not have markdown read them for their function, simply include a backslash (\) before the character, like \$, and it will print out as normal.

Hiding Code

Before exporting notebooks, it is important to place tags into your code cells to hide them; code should not be present in a report as it may make the report much longer than it should be. To hide code cells, click on the cell's setting to add a cell tag, then add "remove-input". If there is a code chunk for which you want to hide the output as well, add "remove-cell".

Now you can export to a PDF. However, it's not nearly that simple. If you try and export a notebook with tagged code cells, the code cells will still produce in the output. Additionally, the required packages for generating PDF output may not be installed on your computer. To fix these, first make sure that the anaconda packages `nbconvert` and its dependencies (`pandoc` and some version of LaTeX) are installed.

- To install `nbconvert`, refer to https://nbconvert.readthedocs.io/en/latest/install.html#installing-nbconvert. This can be done in the terminal with `conda install nbconvert`.
- To install `pandoc`, refer to https://pandoc.org/installing.html. This can be done in the terminal with `brew install pandoc`.
- To install a version of LaTeX, refer to https://tug.org/mactex/ (which installs MacTex for MacOS) or https://miktex.org/ (which installs MikTex for Windows).

Make sure to restart your computer after installing these.

Then following this, make sure you are in the same directory as your notebook that you wish to export, and type the following in the terminal:

```
jupyter nbconvert --to pdf --no-input <file_name>
```

replacing `<file_name>` with the name you want your output file to be. At this point you should be able to produce PDFs from your Jupyter Notebooks!

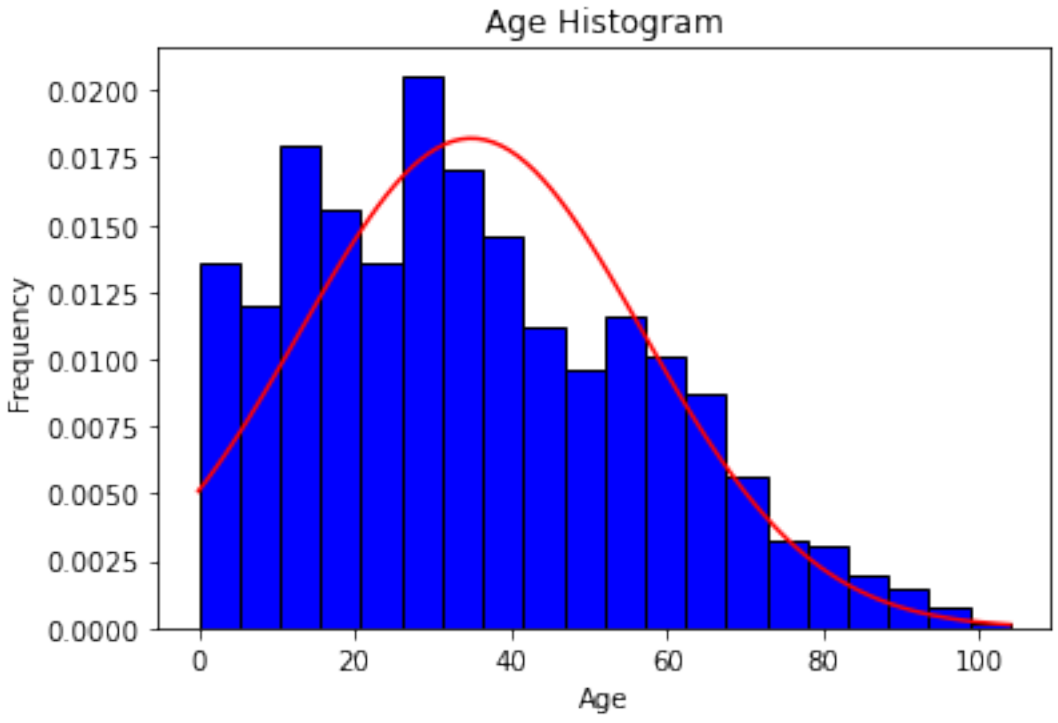
4.8.4 EDA Project, Part 3

Format your findings from the previous sections into a well-written and presentable PDF report with the aforementioned sections.

4.9 Solutions to Part II Exercises

Exercise 7.1

```
#make a histogram for the age column in df
plt.hist(df['age'], bins = 20, color = 'blue', edgecolor = 'black', density=True)

#add normal distribution
mu = df['age'].mean()
sigma = df['age'].std()
x = np.linspace(df['age'].min(), df['age'].max(), 100)
y = 1/(sigma * np.sqrt(2 * np.pi)) * np.exp( - (x - mu)**2 / (2 * sigma**2))

plt.plot(x, y, color = 'red')

plt.title('Age Histogram')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



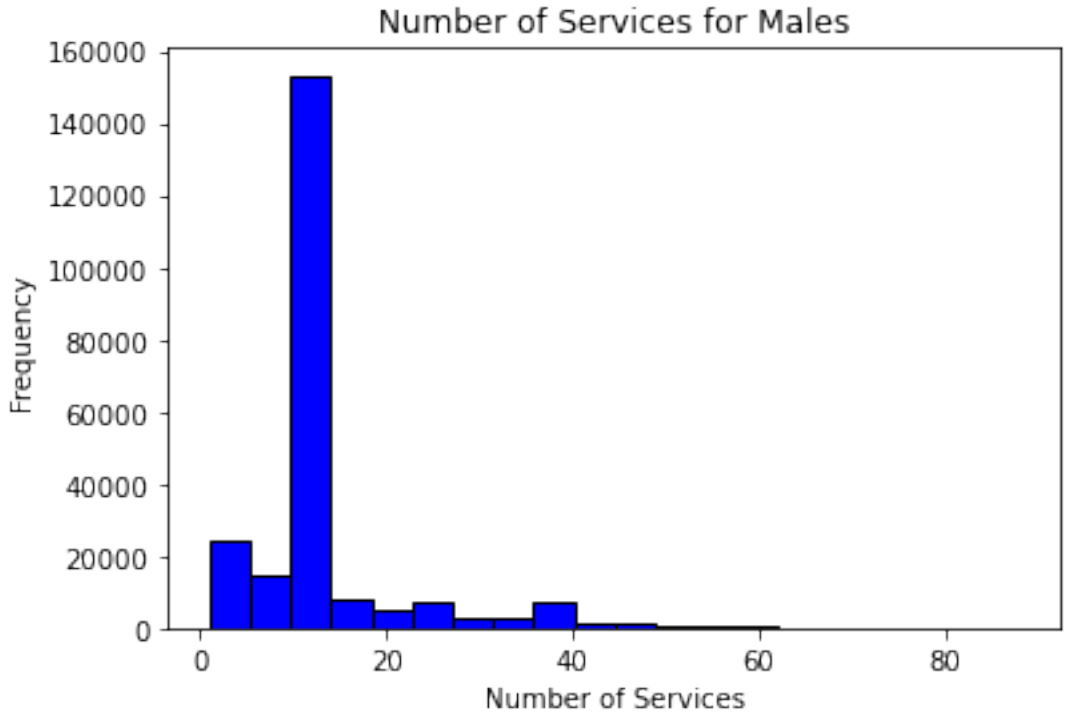
Exercise 7.2

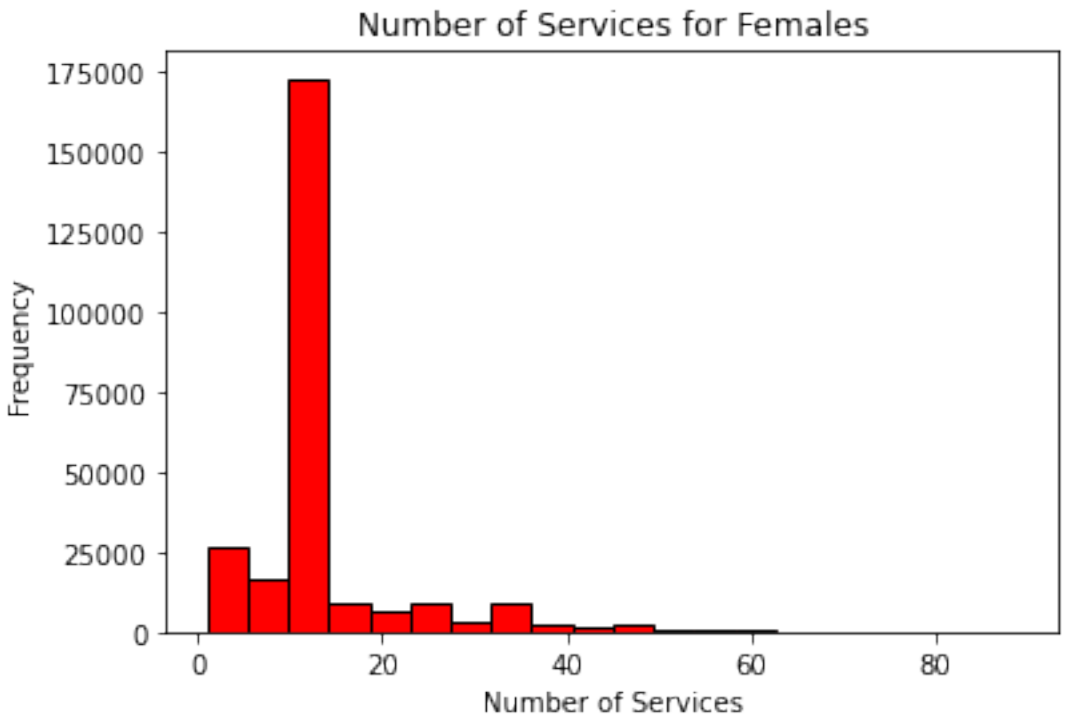
```
#removing "other" entries in gender
intermediate_client_atts = client_attributes[client_attributes['GENDER'] != '99~Other
```

(continues on next page)

(continued from previous page)

```
 ↪']

m = intermediate_client_atts[intermediate_client_atts['GENDER'] == '1~Male']['num_
 ↪services']
f = intermediate_client_atts[intermediate_client_atts['GENDER'] == '2~Female']['num_
 ↪services']

#make a separate histogram for both m and f
plt.hist(m, bins = 20, color = 'blue', edgecolor = 'black')
plt.title('Number of Services for Males')
plt.xlabel('Number of Services')
plt.ylabel('Frequency')
plt.show()

plt.hist(f, bins = 20, color = 'red', edgecolor = 'black')
plt.title('Number of Services for Females')
plt.xlabel('Number of Services')
plt.ylabel('Frequency')
plt.show()
```





Interpretations may vary. While some students may say these look normal given a large number of observations tending around the mean, others may say that due to large-valued outliers, these distributions are not normal. Students are encouraged to perform follow-up statistical Shapiro-Wilk tests.

EDA Project

One example of an EDA report can be found at this link.

https://drive.google.com/file/d/1d-LnndIL3gS9dJbKraxRWPv-fEgNcfkB/view?usp=sharing

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import folium # map rendering library
from folium.features import DivIcon #used to add popup info to a map
import shapely
from shapely.geometry import Point, Polygon
import shapefile # install pyshp first if you get a ModuleNotFoundError
import sklearn
from sklearn.cluster import KMeans
```

4.10 JNB Lab: Housing Equity Initiative

4.10.1 Introduction

In Fall 2021, the Housing Equity Initiative (HEI) was launched by Sunshine Gospel Ministries in partnership with The BLOCK Movement, both non-profit organizations deeply rooted in the South Side of Chicago. Housing inequity between White and Black people living in the United States is a nationwide problem. In 2020 the National Association of Realtors indicated over 70% of Whites live in their own homes, while for Blacks it is less than 45%. Based on Federal Reserve data for 2019, the real estate wealth per capita for Whites is over 140,000, but for Blacks, it is under $60,000.

In the aftermath of the Great Migration of those escaping the Jim Crow South, which brought a large influx of Blacks into Chicago in the first half of the 20th century, decades of housing-related injustices including redlining, blockbusting, restrictive covenants, and slum land-lording have occurred on the South and West sides [Amdat 2021]. Many communities affected by structural racism are carrying the burden of inequitable housing. For example, for the Woodlawn community area with roughly 9,000 housing units, almost 80% are renter-occupied, 36% of which in 2014 were using more than 50% of their income on rent. (For Woodlawn, the 2014 area median income (AMI) was roughly $25,000.)

HEI is focused on affordable home ownership in the Woodlawn area, the active construction site of the $500+ million dollar Obama Presidential Center. HEI seeks to overcome a long history of structural barriers to home ownership by providing three forms of support for vulnerable families:

Tax-Sale Benevolence Program (TSBP) to help existing homeowners avoid foreclosure due to tax sales;

Downpayment Assistance Program (DAP) to assist responsible families with down payment on a home purchase; and

Affordable Revitalization Program (ARP) to revitalize empty lots and rehab properties with well-constructed, owner-occupied homes.

HEI's goal is not simply to increase home ownership for neighborhood residents who have experienced housing inequity, but to maximize positive, rather than negative, home ownership experiences

4.10.2 Park district facilities in Ward 20

There are several factors which can make a difference in whether a young person growing up in a disadvantaged urban community will flourish. One type of community asset are park district facilities. Here we will create a map of such resources for HEI's neighborhood.

1) Import Ward 20 map.

```
img = Image.open("ward20.png")
```

2) Define a function to convert (lat,lon) to an (x,y) location on the Ward 20 map.

```
#Function to convert (lat,lon) to (x,y) location on the Ward 20 map
def coord(lat,lon):
    minlon=-87.672079
    maxlon=-87.586493
    xmin=47
    xmax=806
    x=xmin+(lon-minlon)*(xmax-xmin)/(maxlon-minlon)
    minlat=41.778798
    maxlat=41.801243
    ymax=189
    ymin=389
    y=ymin+(lat-minlat)*(ymax-ymin)/(maxlat-minlat)
    return x,y
```

3) Import a shape file with the Ward 20 boundary.

```
sf = shapefile.Reader("ward20.shp")
shapes=sf.shapes()
```

4) Read in the Chicago Park District data from the Chicago Data Portal.

```
parks=pd.read_json("https://data.cityofchicago.org/resource/eix4-gf83.json?
↪$limit=400000")
```

```
parks.head(1)
```

```
                                          the_geom  objectid_1  \
0  {'type': 'Point', 'coordinates': [-87.63769762...           1

                   park  park_no       facility_n facility_t    x_coord  \
0  HAMILTON (ALEXANDER)         9  CULTURAL CENTER    SPECIAL -87.637698

     y_coord  gisobjid
0  41.762999      2494
```

```
p=parks[["facility_n","facility_t","x_coord","y_coord"]]
p.columns=["type","loc","longitude","latitude"]
p.head(3)
```

```
                  type      loc  longitude   latitude
0       CULTURAL CENTER  SPECIAL -87.637698  41.762999
1             GYMNASIUM   INDOOR -87.637929  41.762817
2  BASEBALL JR/SOFTBALL  OUTDOOR -87.636914  41.760849
```

```
p["type"].value_counts()
```

```
type
BASKETBALL BACKBOARD         771
BASEBALL JR/SOFTBALL         537
PLAYGROUND                   517
TENNIS COURT                 507
BASKETBALL COURT             323
                            ...
GOLF COURSE MINIATURE          1
SPORT ROLLER COIRT             1
NATURE PLAY AREA               1
ALFRED CALDWELL LILY POND      1
ART TURF - REGULATION          1
Name: count, Length: 75, dtype: int64
```

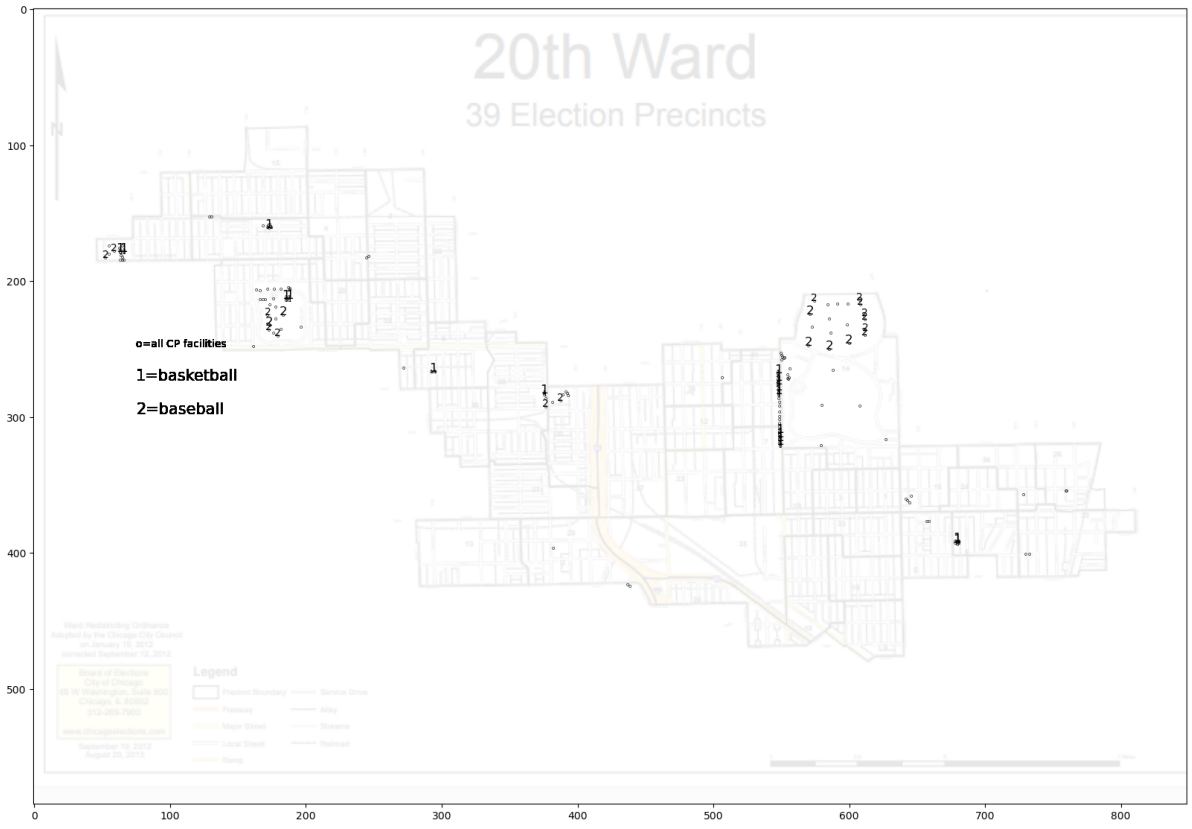
5) Map the Park District data.

```
import matplotlib.pyplot as plt
from PIL import Image
img = Image.open("ward20.png")
plt.figure(figsize=(20,16))
plt.imshow(img,alpha=.1)
df=p

#use this format
```

(continues on next page)

(continued from previous page)

```
for i in df.index:
    [x,y]=coord(df.loc[i,"latitude"],df.loc[i,"longitude"])
    point=Point(df.loc[i,"longitude"],df.loc[i,"latitude"])
    if point.within(Polygon(shapes[3].points)):
        plt.text(x,y,"o",color='black',size=5,ha='center',va='bottom')
        plt.text(75,250,"o=all CP facilities",color='black',size=10,ha='left',va=
↪'bottom')
        if df.loc[i,"type"]=="BASKETBALL BACKBOARD":
            plt.text(x,y,"1",color='black',size=5,ha='center',va='bottom')
            plt.text(75,275,"1=basketball",color='black',size=15,ha='left',va='bottom
↪')
        if df.loc[i,"type"]=="BASKETBALL COURT":
            plt.text(x,y,"1",color='black',size=12,ha='center',va='bottom')
        if df.loc[i,"type"]=="BASEBALL JR/SOFTBALL":
            plt.text(x,y,"2",color='black',size=10,ha='center',va='bottom')
            plt.text(75,300,"2=baseball",color='black',size=15,ha='left',va='bottom')
        if df.loc[i,"type"]=="BASEBALL SR":
            plt.text(x,y,"2",color='black',size=12,ha='center',va='bottom')
plt.savefig("recreation.png")
plt.show()
```



Problem 1

Add playgrounds to the map in red.

4.10.3 Tax Year 2019 Owner-Occupied Tax Sale Data for Ward 20

1) Import tax-sale data.

```
df=pd.read_excel("Ward20residentialparcels.xlsx")
df2=pd.read_excel("HEIcandidateparcels.xlsx")
```

2) Create map of Ward 20 residential parcels and HEI candidate parcels.

Note

Please visit the web version of this section to engage with this interactive resource.

```
Chicago_map = folium.Map(location=[41.78453, -87.62859],    zoom_start=13,alpha=.1)
for i in np.arange(0,169,1):
    p=[df.loc[i,"latitude"],df.loc[i,"longitude"]]
    folium.Marker(p,icon=DivIcon(
        icon_size=(100,0),
        icon_anchor=(0,8),
        html='<div style="font-size: 2pt; color : lightgray">'+'</div>',
            )).add_to(Chicago_map)
    Chicago_map.add_child(folium.CircleMarker(p, radius=1,color='lightgray'))
for i in np.arange(0,92,1):
    p2=[df2.loc[i,"latitude"],df2.loc[i,"longitude"]]
    folium.Marker(p2,icon=DivIcon(
        icon_size=(100,0),
        icon_anchor=(0,8),
        html='<div style="font-size: 6pt; color : black">'+'    '+str(df2.loc[i,"Total␣
 ↪Tax Due"])+ '</div>',
            )).add_to(Chicago_map)
    Chicago_map.add_child(folium.CircleMarker(p2, radius=1,color='black'))
Chicago_map.save("HEItaxsaleyear19maprev.html")
Chicago_map
```

```
<folium.folium.Map at 0x12ecc4860>
```

Exercise

Make a histogram showing tax sale amounts for Ward 20 residential parcels and HEI candidate parcels.

4.10.4 Low Income Tract Clustering

1) Read in census tract data.

```
rawdf=pd.read_csv("tract_covariates.csv")
```

```
rawdf.columns
```

```
Index(['tract', 'county', 'state', 'hhinc_mean2000', 'mean_commutetime2000',
       'frac_coll_plus2010', 'frac_coll_plus2000', 'foreign_share2010',
       'med_hhinc2016', 'med_hhinc1990', 'popdensity2000', 'poor_share2010',
       'poor_share2000', 'poor_share1990', 'share_black2010', 'share_hisp2010',
```

(continues on next page)

(continued from previous page)

```
       'share_asian2010', 'share_black2000', 'share_white2000',
       'share_hisp2000', 'share_asian2000', 'gsmn_math_g3_2013',
       'rent_twobed2015', 'singleparent_share2010', 'singleparent_share1990',
       'singleparent_share2000', 'traveltime15_2010', 'emp2000',
       'mail_return_rate2010', 'ln_wage_growth_hs_grad', 'jobs_total_5mi_2015',
       'jobs_highpay_5mi_2015', 'nonwhite_share2010', 'popdensity2010', 'cz',
       'czname', 'ann_avg_job_growth_2004_2013', 'job_density_2013'],
      dtype='object')
```

```
rawdf.shape
```

```
(74123, 38)
```

2) Filter data to Cook County tracts with median 2016 household income <30,000.

```
IL=rawdf[rawdf['state']== 17]
IL.shape
```

```
(3128, 38)
```

```
cook=IL[IL['county']==31]
cook.shape
```

```
(1319, 38)
```

```
low_inc=cook[cook['med_hhinc2016']<30000]
low_inc.shape
```

```
(202, 38)
```

3) Prepare columns used for separation.

```
df=low_inc[['tract','emp2000', 'frac_coll_plus2010','job_density_2013',   'mean_
↪commutetime2000', 'med_hhinc2016','popdensity2010', 'rent_twobed2015','singleparent_
↪share2010']]
df.columns
```

```
Index(['tract', 'emp2000', 'frac_coll_plus2010', 'job_density_2013',
       'mean_commutetime2000', 'med_hhinc2016', 'popdensity2010',
       'rent_twobed2015', 'singleparent_share2010'],
      dtype='object')
```

```
df.columns=['Tract','emp','college','jobdensity','commute','hhincome','popdensity',
↪'rent','singleparent']
df.head(1)
```

```
       Tract       emp   college  jobdensity    commute  hhincome  popdensity  \
21107  10100  0.560484  0.349921   2530.6123  41.525024   29861.0   33020.406

         rent  singleparent
21107  1153.0      0.543056
```

```
df.loc[:, 'work'] = df.loc[:, 'jobdensity'] / df.loc[:, 'popdensity']
df.loc[:, 'room'] = df.loc[:, 'rent'] / df.loc[:, 'hhincome']
```

4) Normalize values

```
df.shape
```

```
(202, 11)
```

```
df=df.dropna()
df.shape
```

```
(167, 11)
```

```
Mwork=df["work"].max()
mwork=df["work"].min()
Mroom=df["room"].max()
mroom=df["room"].min()
Msingleparent=df["singleparent"].max()
msingleparent=df["singleparent"].min()
Mcommute=df["commute"].max()
mcommute=df["commute"].min()
Mcollege=df["college"].max()
mcollege=df["college"].min()
Mhhincome=df["hhincome"].max()
mhhincome=df["hhincome"].min()
```

```
#normalize values
df.loc[:,"work"]=(df.loc[:,"work"]-mwork)/(Mwork-mwork)
df.loc[:,"room"]=(df.loc[:,"room"]-mroom)/(Mroom-mroom)
df.loc[:,"singleparent"]=(df.loc[:,"singleparent"]-msingleparent)/(Msingleparent-
↪msingleparent)
df.loc[:,"commute"]=(df.loc[:,"commute"]-mcommute)/(Mcommute-mcommute)
df.loc[:,"education"]=(df.loc[:,"college"]-mcollege)/(Mcollege-mcollege)
df.loc[:,"income"]=(df.loc[:,"hhincome"]-mhhincome)/(Mhhincome-mhhincome)
```

```
tracts=df[["Tract","work","room","education","income","singleparent","commute"]]
tracts.shape
```

```
(167, 7)
```

```
tracts=tracts.dropna()
tracts.shape
```

```
(167, 7)
```

5) Use k-means to separate the census tracts into two clusters.

```
cols=["work","room","education","income","singleparent","commute"]
tractcluster=tracts[cols]
```

```
# Fit the k means model
k_means = KMeans(init="k-means++", n_clusters=2, n_init=2)
```

(continues on next page)

(continued from previous page)

```
k_means.fit(tractcluster)
#Get Labels
k_means_labels = k_means.labels_
k_means_labels
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1,
       0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0,
       1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1], dtype=int32)
```

```
tracts["CLASS"]=k_means_labels
tracts=tracts.reset_index(drop=True)
tracts.head(2)
```

```
   Tract      work      room  education    income  singleparent   commute  \
0  10100  0.012277  0.228804   0.423649  0.994194      0.491649  0.554853
1  10202  0.043318  0.222477   0.280312  0.973262      0.617686  0.401514

   CLASS
0      1
1      1
```

```
C0=tracts[tracts["CLASS"]==0]
C0.head(1)
```

```
     Tract      work    room  education    income  singleparent   commute  \
11  231200  0.012669  0.1637   0.129681  0.843523         0.733  0.707592

    CLASS
11      0
```

```
C1=tracts[tracts["CLASS"]==1]
C1.head(1)
```

```
   Tract      work      room  education    income  singleparent   commute  \
0  10100  0.012277  0.228804   0.423649  0.994194      0.491649  0.554853

   CLASS
0      1
```

```
C0.shape
```

```
(96, 8)
```

```
C1.shape
```

```
(71, 8)
```

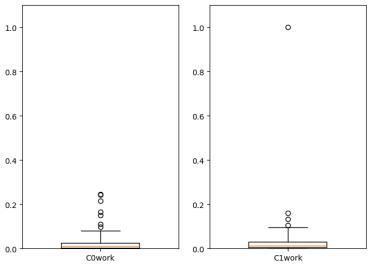
6) Make a boxplot comparison of the two groups

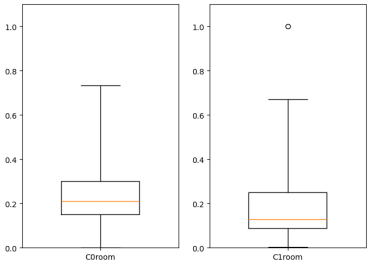
Note

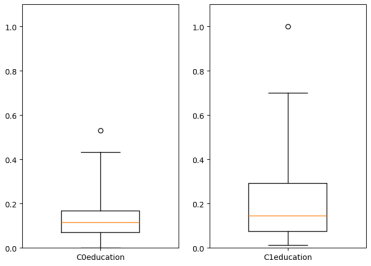
Please visit the web version of this section to engage with this interactive resource.

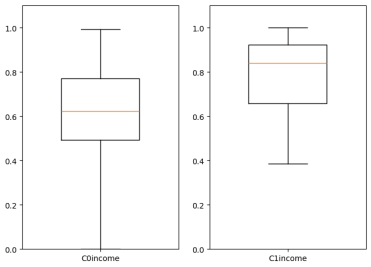
```
numplots=len(cols)
plt.figure(figsize=[10,numplots])
fig, axes = plt.subplots(numplots,2,figsize=[8,40])
for i in np.arange(0,numplots,1):
    axes[i,0].set_ylim((0,1.1))
    axes[i,1].set_ylim((0,1.1))
    axes[i,0].boxplot(C0[cols[i]],whis=3,labels=['C0'+cols[i]],widths=.5)
    axes[i,1].boxplot(C1[cols[i]],whis=3,labels=['C1'+cols[i]],widths=.5)
fig.savefig('6IndR1.png') #Save our figure to a file
plt.show()
```

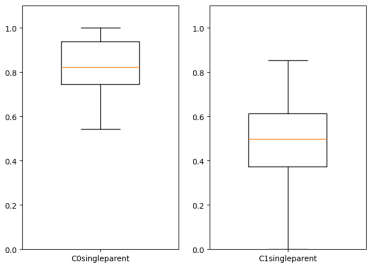
```
<Figure size 1000x600 with 0 Axes>
```

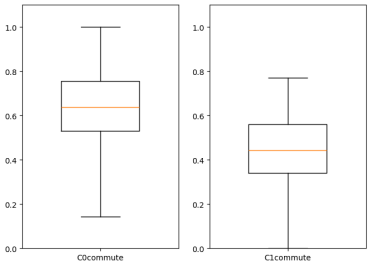

C0work
C1work


C0room
C1room


C0education
C1education


C0income
C1income


C0singleparent
C1singleparent


C0commute
C1commute

```
df2.shape
```

```
(92, 34)
```

Exercise

Continue the same process one further step starting with the sub-cluster (C0 or C1) that exhibits greater hardship.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import folium # map rendering library
from folium.features import DivIcon #used to add popup info to a map
import shapely
from shapely.geometry import Point, Polygon
import shapefile # install pyshp first if you get a ModuleNotFoundError
import sklearn
from sklearn.cluster import KMeans
```

4.11 JNB Lab Solutions

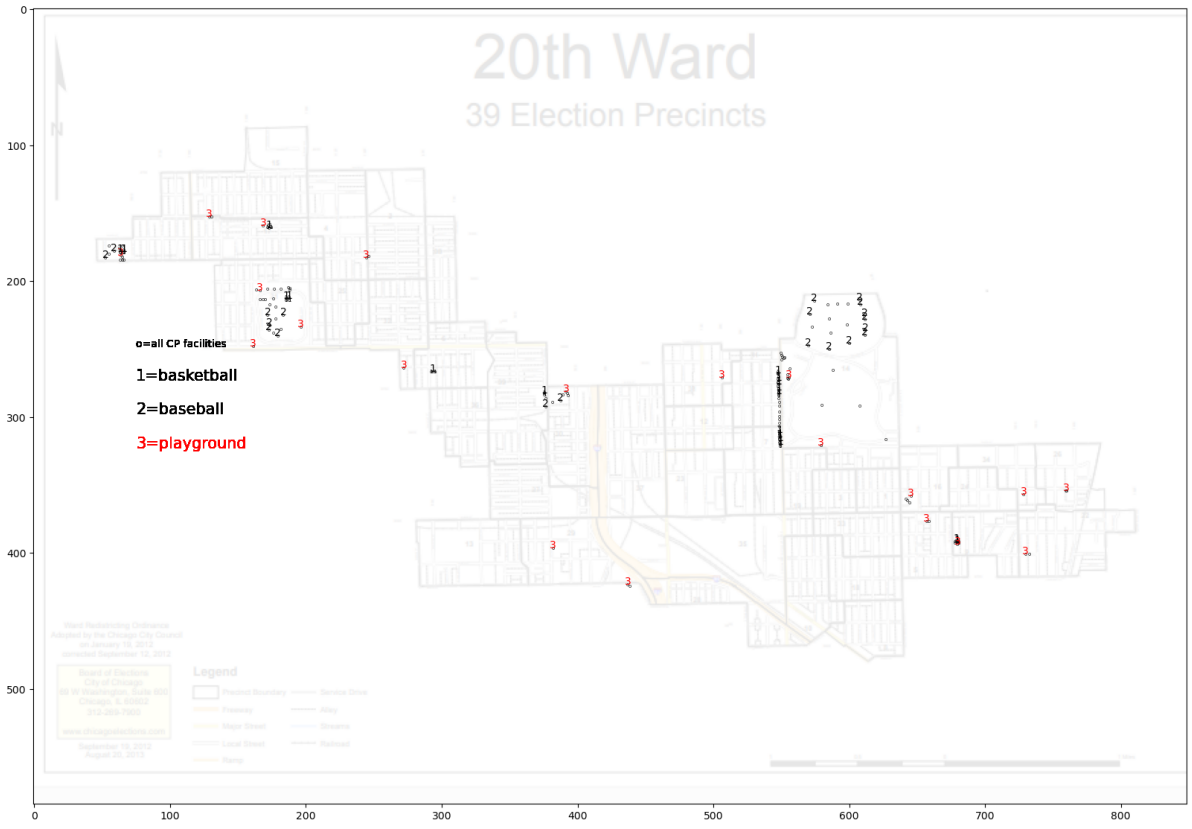
4.11.1 Park District Facilities

```
img = Image.open("ward20.png")
#Function to convert (lat,lon) to (x,y) location on the Ward 20 map
def coord(lat,lon):
    minlon=-87.672079
    maxlon=-87.586493
    xmin=47
    xmax=806
    x=xmin+(lon-minlon)*(xmax-xmin)/(maxlon-minlon)
    minlat=41.778798
    maxlat=41.801243
    ymax=189
    ymin=389
    y=ymin+(lat-minlat)*(ymax-ymin)/(maxlat-minlat)
    return x,y
sf = shapefile.Reader("ward20.shp")
shapes=sf.shapes()
parks=pd.read_json("https://data.cityofchicago.org/resource/eix4-gf83.json?
 ↪$limit=400000")
p=parks[["facility_n","facility_t","x_coord","y_coord"]]
p.columns=["type","loc","longitude","latitude"]
plt.figure(figsize=(20,16))
plt.imshow(img,alpha=.1)
df=p

#use this format
for i in df.index:
    [x,y]=coord(df.loc[i,"latitude"],df.loc[i,"longitude"])
```

(continues on next page)

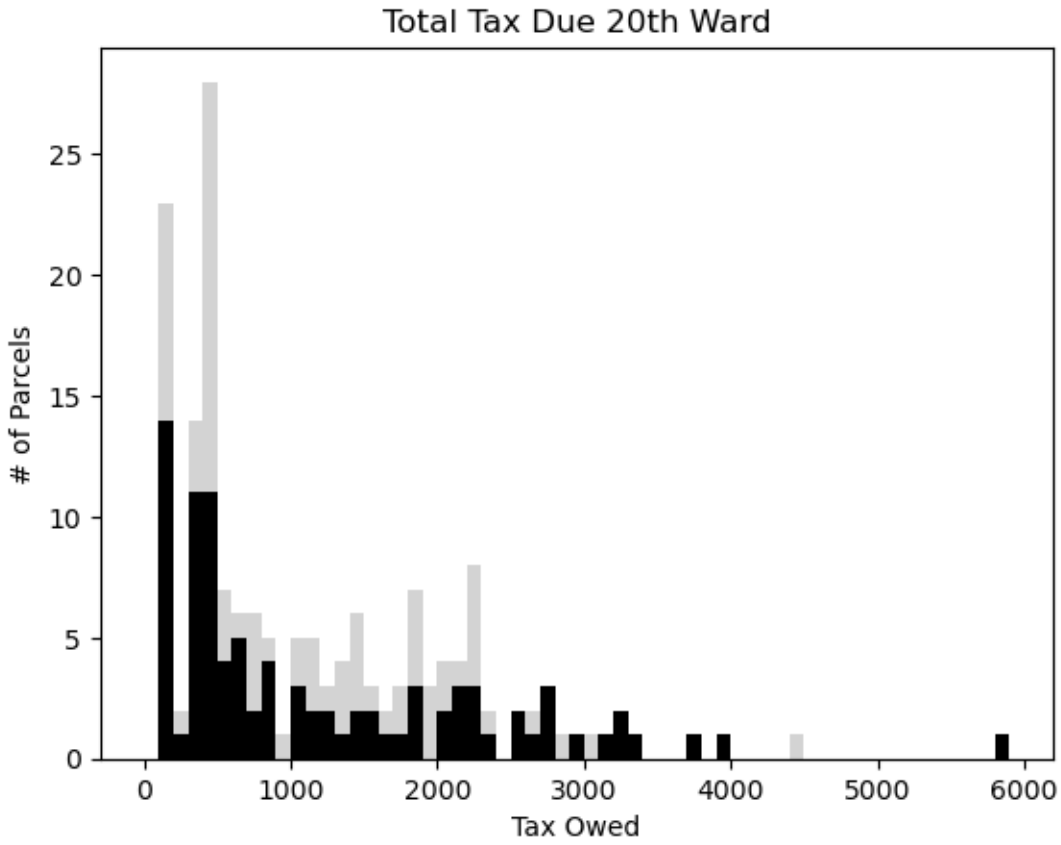
(continued from previous page)

```
    point=Point(df.loc[i,"longitude"],df.loc[i,"latitude"])
    if point.within(Polygon(shapes[3].points)):
        plt.text(x,y,"o",color='black',size=5,ha='center',va='bottom')
        plt.text(75,250,"o=all CP facilities",color='black',size=10,ha='left',va=
↪'bottom')
        if df.loc[i,"type"]=="BASKETBALL BACKBOARD":
            plt.text(x,y,"1",color='black',size=5,ha='center',va='bottom')
            plt.text(75,275,"1=basketball",color='black',size=15,ha='left',va='bottom
↪')
        if df.loc[i,"type"]=="BASKETBALL COURT":
            plt.text(x,y,"1",color='black',size=10,ha='center',va='bottom')
        if df.loc[i,"type"]=="BASEBALL JR/SOFTBALL":
            plt.text(x,y,"2",color='black',size=10,ha='center',va='bottom')
            plt.text(75,300,"2=baseball",color='black',size=15,ha='left',va='bottom')
        if df.loc[i,"type"]=="BASEBALL SR":
            plt.text(x,y,"2",color='black',size=10,ha='center',va='bottom')
        if df.loc[i,"type"]=="PLAYGROUND":
            plt.text(x,y,"3",color='red',size=10,ha='center',va='bottom')
            plt.text(75,325,"3=playground",color='red',size=15,ha='left',va='bottom')
plt.show()
```



4.11.2 Tax Year 2019 Owner-Occupied Tax Sale Data for Ward 20

```
df=pd.read_excel("Ward20residentialparcels.xlsx")
df2=pd.read_excel("HEIcandidateparcels.xlsx")
# plotting first histogram
df["Total Tax Due"].hist(grid=False, bins=range(0,6000,100),color='lightgray')
# plotting second histogram
df2["Total Tax Due"].hist(grid=False, bins=range(0,6000,100),color='black')
plt.xlabel('Tax Owed')
plt.ylabel('# of Parcels')

plt.title('Total Tax Due 20th Ward')
plt.savefig("TS2019rev.png")
# Showing the plot using plt.show()
plt.show()
```



4.11.3 Low Income Tract Clustering

Note

Please visit the web version of this section to engage with this interactive resource.

3a) Read in census tract data.

```
rawdf=pd.read_csv("tract_covariates.csv")
IL=rawdf[rawdf['state']== 17]
cook=IL[IL['county']==31]
low_inc=cook[cook['med_hhinc2016']<30000]
df=low_inc[['tract','emp2000', 'frac_coll_plus2010','job_density_2013',   'mean_
 ↪commutetime2000', 'med_hhinc2016','popdensity2010', 'rent_twobed2015','singleparent_
 ↪share2010']]
df.columns=['Tract','emp','college','jobdensity','commute','hhincome','popdensity',
 ↪'rent','singleparent']
df.loc[:, 'work'] = df.loc[:, 'jobdensity'] / df.loc[:, 'popdensity']
df.loc[:, 'room'] = df.loc[:, 'rent'] / df.loc[:, 'hhincome']
df=df.dropna()
 # normalize values
Mwork=df["work"].max()
mwork=df["work"].min()
Mroom=df["room"].max()
mroom=df["room"].min()
Msingleparent=df["singleparent"].max()
msingleparent=df["singleparent"].min()
Mcommute=df["commute"].max()
mcommute=df["commute"].min()
Mcollege=df["college"].max()
mcollege=df["college"].min()
Mhhincome=df["hhincome"].max()
mhhincome=df["hhincome"].min()
df.loc[:,"work"]=(df.loc[:,"work"]-mwork)/(Mwork-mwork)
df.loc[:,"room"]=(df.loc[:,"room"]-mroom)/(Mroom-mroom)
df.loc[:,"singleparent"]=(df.loc[:,"singleparent"]-msingleparent)/(Msingleparent-
 ↪msingleparent)
df.loc[:,"commute"]=(df.loc[:,"commute"]-mcommute)/(Mcommute-mcommute)
df.loc[:,"education"]=(df.loc[:,"college"]-mcollege)/(Mcollege-mcollege)
df.loc[:,"income"]=(df.loc[:,"hhincome"]-mhhincome)/(Mhhincome-mhhincome)
```

```
C:\Users\pisihara\AppData\Local\Temp\ipykernel_22240\83952099.py:7:␣
 ↪SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/
 ↪user_guide/indexing.html#returning-a-view-versus-a-copy
  df.loc[:, 'work'] = df.loc[:, 'jobdensity'] / df.loc[:, 'popdensity']
C:\Users\pisihara\AppData\Local\Temp\ipykernel_22240\83952099.py:8:␣
 ↪SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/
```

(continues on next page)

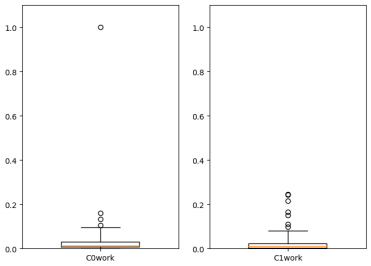
(continued from previous page)

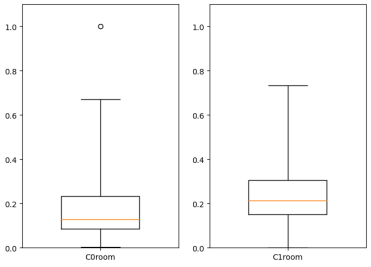
```
↪user_guide/indexing.html#returning-a-view-versus-a-copy
 df.loc[:, 'room'] = df.loc[:, 'rent'] / df.loc[:, 'hhincome']
```

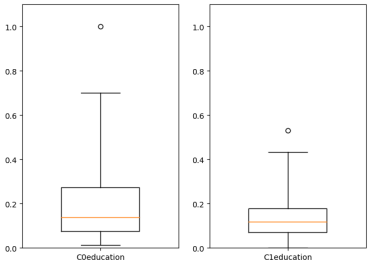
```
tracts=df[["Tract","work","room","education","income","singleparent","commute"]]
tracts=tracts.dropna()
cols=["work","room","education","income","singleparent","commute"]
tractcluster=tracts[cols]
# Fit the k means model
k_means = KMeans(init="k-means++", n_clusters=2, n_init=2)
k_means.fit(tractcluster)
#Get Labels
k_means_labels = k_means.labels_
k_means_labels
tracts["CLASS"]=k_means_labels
tracts=tracts.reset_index(drop=True)
tracts.head(2)
C0=tracts[tracts["CLASS"]==0]
C0.head(1)
C1=tracts[tracts["CLASS"]==1]
C1.head(1)
numplots=len(cols)
plt.figure(figsize=[10,numplots])
fig, axes = plt.subplots(numplots,2,figsize=[8,40])
for i in np.arange(0,numplots,1):
    axes[i,0].set_ylim((0,1.1))
    axes[i,1].set_ylim((0,1.1))
    axes[i,0].boxplot(C0[cols[i]],whis=3,labels=['C0'+cols[i]],widths=.5)
    axes[i,1].boxplot(C1[cols[i]],whis=3,labels=['C1'+cols[i]],widths=.5)
plt.show()
```

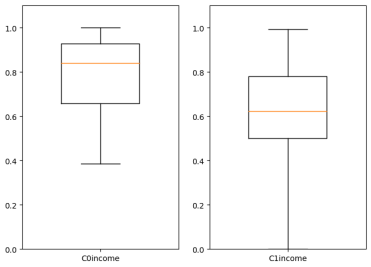
```
C:\Users\pisihara\AppData\Local\anaconda3\Lib\site-packages\sklearn\cluster\_
↪kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows␣
↪with MKL, when there are less chunks than available threads. You can avoid it by␣
↪setting the environment variable OMP_NUM_THREADS=1.
 warnings.warn(
```

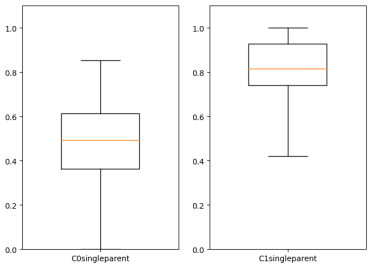
```
<Figure size 1000x600 with 0 Axes>
```

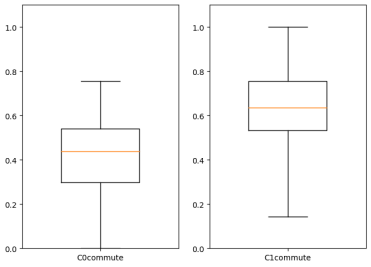

C0work
C1work


C0room
C1room


C0education
C1education


C0income
C1income

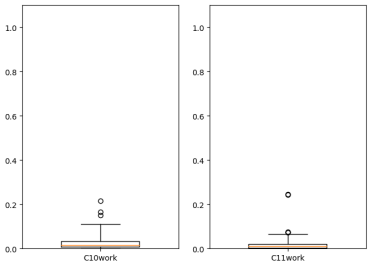

C0singleparent
C1singleparent


C0commute
C1commute

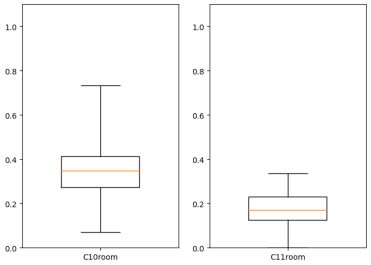
```
tracts=C1[["Tract","work","room","education","income","singleparent","commute"]]
tracts=tracts.dropna()
cols=["work","room","education","income","singleparent","commute"]
tractcluster=tracts[cols]
# Fit the k means model
k_means = KMeans(init="k-means++", n_clusters=2, n_init=2)
k_means.fit(tractcluster)
#Get Labels
k_means_labels = k_means.labels_
k_means_labels
tracts["CLASS"]=k_means_labels
tracts=tracts.reset_index(drop=True)
tracts.head(2)
C10=tracts[tracts["CLASS"]==0]
C11=tracts[tracts["CLASS"]==1]
numplots=len(cols)
plt.figure(figsize=[10,numplots])
fig, axes = plt.subplots(numplots,2,figsize=[8,40])
for i in np.arange(0,numplots,1):
    axes[i,0].set_ylim((0,1.1))
    axes[i,1].set_ylim((0,1.1))
    axes[i,0].boxplot(C10[cols[i]],whis=3,labels=['C10'+cols[i]],widths=.5)
    axes[i,1].boxplot(C11[cols[i]],whis=3,labels=['C11'+cols[i]],widths=.5)
plt.show()
```

```
C:\Users\pisihara\AppData\Local\anaconda3\Lib\site-packages\sklearn\cluster\_
 ↪kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows␣
 ↪with MKL, when there are less chunks than available threads. You can avoid it by␣
 ↪setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

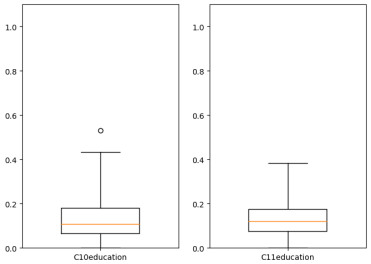
```
<Figure size 1000x600 with 0 Axes>
```


C10work

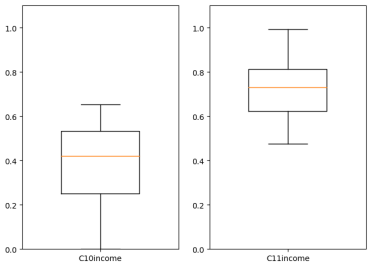
C11work


C10room

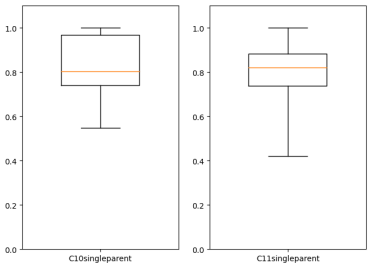
C11room


C10education

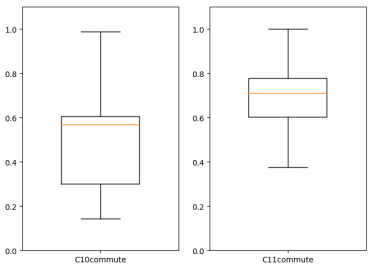
C11education


C10income

C11income


C10singleparent

C11singleparent


C10commute

C11commute

CHAPTER 5

PROBABILITY - Laura Gross, Yiheng Liang

Note

Dr. Gross (Computer Science and Mathematics, Bridgewater State University) and Dr. Liang (Computer Science, University of Georgia) will explain probability models in conjunction with simulations.

5.1 Introduction

The **probability** is the chance or likelihood that a given event will occur. For example, we can ask, "What is the probability that it will rain tomorrow?", and evaluate the possibility of that event occurring mathematically. Probability is a branch of mathematics concerned with the study of probabilities.

We quantify the probabilities of various events in this chapter. We do both mathematical analysis and computation. As in other fields of mathematics, mathematical understanding guides the choices of what to compute, and insights gained from computational exploration enhance mathematical understanding.

Ultimately we use **models** to describe practical problems via mathematics or an **algorithm** (precise method). We solve mathematical models approximately using Python programs or implement algorithms using Python. The results mimic, imitate, or simulate the behavior of the physical or practical problem at hand. We can use Python to create visualizations that facilitate interpretation of the computer simulation.

5.2 Random Numbers

Many areas of study involve random numbers. Examples include probability, computational science, data science, and statistics.

5.2.1 Arbitrariness

When you tell a friend to "Pick a number between 1 and 10," you may think you are telling the friend to pick a **random** number. However, for true randomness, each outcome 1, 2, 3, …, 10 must be equally likely. It turns out that people have a preference, usually unconscious, for picking some numbers over others. Therefore, you are really telling the friend to pick an **arbitrary** integer between 1 and 10, rather than a random one.

5.2.2 Pseudo-randomness

Even when computers do the choosing, most of time we should only say we have generated pseudo-random (fake random) numbers. A pseudo-random number generator, or PRNG, is a program that uses mathematical methods to **simulate** (mimic) randomness.

The basic process is to:

1. Provide the PRNG with an arbitrary seed (starting condition).
2. Ask the PRNG for the next "random" number. It is produced by a step-by-step, complicated, but not random, process.

Such a generator is also called a **deterministic** random bit generator. "Deterministic" means causally determined by preceding actions. The word "deterministic" is typically used in contrast to "random."

Sometimes, researchers use a cryptographically secure pseudo-random number generator (CSPRNG). A CSPRNG may sound like it uses true randomness (for strong security). However, you cannot safely assume that you have a completely fair (unbiased) "coin flip" or means of selection, even in this context. Fortunately, though, the prescriptive (deterministic) mechanism involved is hard to reverse, ensuring some measure of security. The key is to have a CSPRNG that is good enough for the application at hand.

5.2.3 Pseudo-random number generation in Python

People typically speak of computers **generating random numbers**. We will use this language when we use Python. However, Python (or Excel, Java, C, C++, or R) is technically doing pseudo-random number generation.

In Python, a default mechanism is used to set the seed for the PRNG. If you want to use the default mechanism explicitly, you can import the`random`library and do

```
import random
random.seed()
```

If for any reason you want to set the seed to a specific integer, such as 123, you can do

```
random.seed(123)
```

5.2.4 True randomness

A truly random number cannot simply be generated from a certain **algorithm** (detailed method) or by a computer itself (unless the computer is purposely re-designed). Truly-random number generation is possible when we consider some physical phenomenon or process, such as atmospheric noise (static from lightning discharges), cosmic background radiation, etc. While truly random numbers are useful in many practical fields, it is sufficient for us to generate and use pseudo-random numbers in this book.

5.2.5 Exercises

Exercises

Exercise 1. Which statement do you judge to be more accurate? Why?

1. The computer program selected a number between 1 and 10 at random.
2. The computer program selected a number between 1 and 10 arbitrarily.

Exercise 2. When a computer program selects a number between 1 and 10, do you think that the choice involves possible preference for some number over others? Support your answer with examples.

Exercise 3: Think of an example of how randomness might be involved in a study investigating a new medicine. Explain how randomness is involved. Could computation, for example using Python, be helpful in the study? Explain your answer.

Exercise 4: Think of an example of how randomness might occur in the natural world. Explain how randomness is involved. Could a computer simulate or mimic a natural process in the context of your example? If so, how could a computer simulation be useful?

Exercise 5: Think of an example of how randomness might occur in a game. Explain how randomness is involved. Could a computer simulate the game? If so, how could the computer simulation be useful?

Exercise 6: Consider a die (plural: “dice”). (See the image below.) Do you think that when we roll a physical die, we can be certain that there is an equal chance of the die landing with each of the six sides facing up? Or could there be a bias toward some sides? Explain your thinking. Do you think the outcome of a roll of the physical die could be influenced by the previous roll? Explain your thinking. How could we use a computer program to simulate a die roll? What are some advantages of doing so?

Exercise 7: Consider the following answer to the question "True or false— An arbitrary choice is made for no special reason":

"Yes, an arbitrary choice typically refers to a decision or selection that is made without any specific reason or justification. It means that there is no objective or rational basis for choosing one option over another. Arbitrary choices are often made when there is no clear preference or when the decision doesn't have a significant impact on the outcome. In such cases, people may resort to random selection, personal whim, or other non-logical factors to make a choice." (ChatGPT, personal communication, June 6, 2023).

Provide a critique for which parts are right (if any). Provide justification for why they are correct. Identify which parts are wrong (if any) and explain what is wrong. Rewrite the answer in your own words with any corrections that are needed. (Retain the citation as with any paraphrased material.) Add an example that you think of yourself to clarify the answer to the true/false question.

Exercise 8: Use the code below to explore Python commands in the`random`library. Read the documentation here as desired: https://docs.python.org/3/library/random.html . For each`print`statement, insert a comment (using`#`) that describes all the possible numbers that can be generated. We see possible output below.

Exercise 9: Which does a Python program generate: a pseudo-random number or a truly random number? Give an example, and explain why the example is pseudo random or truly random.

```
print(random.randrange(6))
print(random.randrange(6) + 1)
print(random.randrange(1, 7))
print(random.randint(1, 6))
print(random.randrange(0, 10, 2))
print(random.uniform(10, 20))
```

```
0
3
1
```

(continues on next page)

(continued from previous page)

```
4
4
11.07700234938439
```

```
import random
```

5.3 Probability

Consider a six-sided die. (See image of two six-sided dice in the previous section.) The die can be used to select a number between 1 and 6 inclusive. By **inclusive**, we mean inclusive of the "end points" 1 and 6 (as well as of 2, 3, 4, and 5). The die can be used to select one of the numbers in an approximately random manner, modulo the slight imperfections (physical asymmetries) in the die.

Suppose you roll a perfect die n times and keep a record of each outcome.

We list the following examples: What is the chance or **probability** that…

1. the die lands on a given number (say, 3) on the first roll?
2. the die lands on a given number (say, 4) on the second roll?
3. the die lands on a 1 or 6 on the first roll?
4. the die lands on the same number on the first two rolls?
5. the die lands on 1, 2, 3, 4, 5, and then 6 when rolled six times ($n = 6$)?

The **probability** of an event, given a set of all of the equally likely outcomes, is the ratio

$$\frac{\text{the number of outcomes that produce the event}}{\text{the total number of possible outcomes}}.$$

Does this definition help you answer Example 1? Give it a try. Then read on!

We can do all of Examples 1–5 **approximately** by using **computer simulation**. The computer **simulates** or imitates the rolling of a physical die. We can generate (pseudo-)random numbers many times and see what fraction of the cases result in the target outcome. Also, we can answer Problems 1–5 **exactly** by using **mathematical ideas**.

5.3.1 Definitions and Facts

Definitions and rules. The **sample space** is the set of possible outcomes. The sample space for the roll of the die is {1, 2, 3, 4, 5, 6}.

Definition. An **event** E corresponds to a subset of the sample space. For example, one event E is that the die lands on a 1 or a 6. The event E corresponds to the subset {1, 6}.

Example 1. The probability that the die lands on a given number (say, 3) on the first roll is 1/6.

Fact. The probability of an event A is a real number $P(A)$ between 0 and 1 inclusive, for example 1/6.

Fact. The probability of the sample space is 1. For example, the probability that the die lands on a 1 or 2 or 3 or 4 or 5 or 6 is 6/6 or 1.

Fact. The probability that event A does not occur is $P(\text{not}A) = 1 - P(A)$.

Example 3. The probability that the die lands on a 1 or a 6 is 2/6 or 1/3.

Example 4. The sample space for two rolls of the die is {11, 12,…, 16, 21, 22,…, 26, 31, 32,…, 36,…61, 62,…, 66}, where each two-digit number represents two rolls of the die. An event A, in which the die lands on the same number on the first two rolls, corresponds to the set {11, 22, 33, 44, 55, 66}. The probability that the die lands on the same number on the first two rolls is 6/36 or 1/6.

Definition. **Mutually exclusive** events are events that cannot happen at the same time. For example, one roll of the die can yield one of six mutually exclusive events.

5.3.2 Addition rule

The addition rule for mutually exclusive events A and B is that $P(A \text{ or } B) = P(A) + P(B)$.

Example 3. For example, the probability that the die lands on 1 (event A) or the die lands on 6 (event B) is $P(A \text{ or } B) = P(A) + P(B) = 1/6 + 1/6 = 1/3$. This agrees with the prior solution.

5.3.3 Multiplication rule

The multiplication rule for mutually exclusive events A and B is that $P(A \text{ and } B) = P(A)P(B)$.

Example 4. For example, the probability that the die lands on the same number on the first two rolls is the probability that a number is rolled (event A) and the same number is rolled (event B) is $P(A \text{ and } B) = P(A)P(B) = 1 * 1/6 = 1/6$. This agrees with the prior solution.

5.3.4 Simulation

This section opened with Problems 1–5 above. We said we could use a computer program to **simulate** the rolling of a physical die to answer the questions approximately.

Let's write a `roll` function to generate (pseudo-)random numbers many times. We'll approximate the probability of an event by calculating the fraction of the total cases that result in the event. We see a possible output below.

```
# Return a list of num_rolls die rolls.
def roll(num_rolls):
    # Define a list for the results of the die rolls.
    die_rolls = []
    # Roll the die num_rolls times.
    for i in range(num_rolls):
        # Store a number 1-6 inclusive in die_roll.
        die_roll = random.randint(1, 6)
        # Append the die roll into the list of die rolls.
        die_rolls.append(die_roll)

    return(die_rolls)
```

```
# Store ten rolls of the die in a list.
rolls = roll(10)
# Print the list.
print(rolls)
# Print the number of threes that were rolled.
print(f"Number of threes: {rolls.count(3)}")
```

```
[3, 3, 4, 4, 1, 3, 3, 2, 4, 5]
Number of threes: 4
```

Simulating Example 1. Answer Example 1 from the beginning of this section approximately by simulating a large number of rolls and counting the number of threes. Compare the result to the mathematical answer in a complete sentence. We see a possible output below.

```
For large number of trials, the simulation should yield a fraction of threes␣
 ↪approximately equal to 1/6 or 0.16666666666666666.
Result obtained: 0.16629
```

For 100,000 simulated rolls of the die, the fraction of values of 3 is typically rather close to the mathematical probability of rolling a 3.

Simulating Example 3. Answer Example 3 from the beginning of this section approximately by simulating a large number of rolls and counting the number of ones and sixes. Compare the result to the mathematical answer in a complete sentence. We see a possible output below.

```
For large number of trials, the simulation should yield a fraction approximately␣
 ↪equal to 2/6 or 0.3333333333333333.
Result obtained: 0.336
```

Simulating Example 4. Answer Example 4 from the beginning of this section approximately by doing a large number of trials of two rolls of a die and counting the fraction of times the two rolls are the same. Compare the result to the mathematical probability in a complete sentence.

```
For large number of trials, the simulation should yield a fraction of 'double␣
 ↪numbers' approximately equal to 1/6 or 0.16666666666666666.
Result obtained: 0.16767
```

For 100,000 simulated rolls of the die, the fraction of times the two rolls came up is typically rather close to the mathematical probability of the two rolls coming up the same.

5.3.5 Law of large numbers

In probability theory, the law of large numbers (LLN) says that the average of the results obtained from a large number of trials should be close to the expected average and tends to become closer to the expected average as more trials are performed (based on the Wikipedia article on the LLN, citing A Modern Introduction to Probability and Statistics: Understanding Why and How, by F.M. Dekking, C. Kraaikamp, H.P. Lopuhaa, and L.E. Meester. Springer (Delft, The Netherlands, 2005)).

Example. A single roll of a fair, six-sided die produces 1, 2, 3, 4, 5, or 6, each with equal probability. Therefore, the expected value of the average of the rolls is (1 + 2 + 3 + 4 + 5 + 6) / 6 = 3.5. According to the LLN, if a six-sided die is rolled a large number of times, the average value will approach 3.5 as the number of die rolls approaches infinity. We will illustrate this idea using simulations with a larger and larger amount of rolls. We see a possible output below.

```
for i in range(6):
    n = 10**(i+1)
    print(f"{n}, {sum(roll(n))/n}")
```

```
10, 3.3
100, 3.42
1000, 3.547
10000, 3.5135
100000, 3.49626
1000000, 3.498728
```

For a number of rolls $n = 10, 100, 1000, 10^4, 10^5, 10^6$, the average value is, for example, 3.3, 3.42, 3.547, 3.5135, 3.49626, 3.498728, respectively.

5.3.6 Gambler's fallacy

"…the law of large numbers does not imply, as too many seem to think, that if deviations from expected behavior occur, these deviations are likely to be 'evened out' by opposite deviations in the future. This misapplication of the law of large numbers is known as the gambler's fallacy" (Introduction to Computation and Programming Using Python by Guttag).

Example. At the Casino de Monte-Carlo on August 18, 1913, the roulette ball landed on black 26 times in a row, and gamblers lost millions betting on red, incorrectly believing that landing on red became increasingly likely as the length of the run of blacks increased (https://www.bbc.com/future/article/20150127-why-we-gamble-like-monkeys , 1/27/2015). If you assume that there is a probability of 0.5 of landing on black and a probability of 0.5 of landing on red, what is the probability that 26 spins will result in black all 26 times? What is the probability that the 27th spin will land on black?

```
# If you assume that there is a probability of 0.5 of landing on black and a␣
↪probability of 0.5 of landing on red,
# the probability that 26 spins will result in black all 26 times is
print((0.5)**26)
print(1.5/100_000_000)
# If you do 100 million trials of spinning 26 times each, getting black 26 times will␣
↪typically happen only once or twice.
```

```
1.4901161193847656e-08
1.5e-08
```

If you spin a 27th time, the probability of landing on black is 0.5.

5.3.7 Exercises

Exercises

Exercise 1: Do Example 5 from the beginning of the Probability section mathematically: Suppose you roll a perfect die n times and keep a record of each outcome. What is the chance or probability that the die lands on 1, 2, 3, 4, 5, and then 6 when rolled six times ($n = 6$)? Use Python to evaluate your answer. Explain your conclusion in a complete sentence.

Exercise 2: Do Example 5 from the beginning of the Probability section computationally: Suppose you roll a perfect die n times and keep a record of each outcome. What is the chance or probability that the die lands on 1, 2, 3, 4, 5, and then 6 when rolled six times ($n = 6$)? Do a large number of trials of six rolls of a die and count the number of times the rolls come out as 1 followed by 2, followed by 3, followed by 4, followed by 5, followed by 6. A piece of syntax you may consider using is `if (rolls[0] == 1 and rolls[1] == 2 and rolls[2] == 3 and rolls[3] == 4 and rolls[4] == 5 and rolls[5] == 6):`. Compare the fraction times you get this pattern to your mathematical answer from the previous exercise in a complete sentence.

Exercise 3: If event A is that the die comes up 2, and event B is that the die comes up with an even number, are the two events A and B mutually exclusive? Explain. Is the addition rule $P(A \text{ or } B) = P(A) + P(B)$ applicable? If so, show that the rule holds. If not, show that the rule does not hold. Use Python to illustrate your answer.

Exercise 4: Calculate the probability (a) mathematically and (b) computationally of obtaining the given result in n rolls of a fair six-sided die. Make a statement in a complete sentence about how many times you expect the outcome, in terms of a number of trials (consisting of n rolls per trial). Compare the fraction of times your simulation produces the target pattern with your mathematical answer.

1. $n = 6$, all numbers are the same
2. $n = 6$, at least one number is different
3. $n = 1$, the number rolled is greater than 2

4. $n = 2$, the sum of the two numbers is greater than 2
5. $n = 2$, the two numbers are different
6. $n = 3$, two numbers are the same and one is different
7. $n = 6$, all the numbers are different
8. $n = 100$, all the numbers are the same
9. $n = 6$, the number 1 appears at least once
10. $n = 6$, the number 1 appears at most once

Exercise 5: Consider the function **flip** below. Summarize its purpose in a complete sentence. Add comments (with #) to clarify the code. Demonstrate its performance on a sample run. Summarize the results in a complete sentence.

Exercise 6. Consider the function **flip_trials_preliminary** below. Add comments to explain the code. Summarize the purpose of the function in a complete sentence. Demonstrate its performance on a sample run. Summarize the results in a complete sentence.

Exercise 7. When a fair coin is flipped once, the theoretical probability that the outcome will be heads is equal to 1/2. Therefore, according to the LLN, the proportion of heads in a "large" number n of coin flips "should be" roughly 1⁄2, and the proportion of heads approaches 1/2 as n approaches infinity. Illustrate this idea using simulation with a larger and larger number of flips.

```
def flip(num_flips):
    head_count = 0
    for i in range(num_flips):
        if random.choice(('H', 'T')) == 'H':
            head_count += 1
    return head_count/num_flips
```

```
def flip_trials_preliminary(num_flips_per_trial, num_trials):
    fraction_heads = []
    for i in range(num_trials):
        fraction_heads.append(flip(num_flips_per_trial))
    mean = sum(fraction_heads) / len(fraction_heads)
    return mean
```

5.4 Conditional Probability

Independent events are unrelated events. For example, a die roll coming up 1 is independent of whether the previous roll came up 1.

Many common events are not independent. Consider and discuss the following events and whether the probability of each is high or low:

- your bicycle needs repairs,
- you have a bicycle accident,
- your bicycle needs repairs given that you had a bicycle accident,
- you had a bicycle accident, given that your bicycle needs repairs.

Question: Are the bicycle repairs and the bicycle accident independent events?

If not, we call them **conditionally related**.

Question: Which statement is more logical? Or are they equally logical?

1. Whether your bike needs repairs (event R) is related to whether you have a bike accident (event A).
2. Whether you have a bike accident (event A) is related to whether your bike needs repairs (event R).

The **conditional probability** of an event is based on prior knowledge of conditions that might be related to the event (based on the Wikipedia article on Bayes' Theorem). We speak of the conditional probability that event Y occurs, given that event X occurred. We denote the probability as P(Y|X).

For example, we may want to know the probability that your bike needs repairs (event R) given that you had a bike accident (event A). We denote the probability as P(R|A).

The quantity P(R|A) is a conditional probability. Here A is considered the **prior** event. We say we are **conditioning on** A.

The (conditional) probability of Y given X can be calculated using the mathematical **definition**

$$P(Y|X) = \frac{P(X \text{ and } Y)}{P(X)}.$$

The probability P(X) of the prior event X is called the prior probability.

We can derive **Bayes' Theorem** by first writing the conditional probability of X given Y

$$P(X|Y) = \frac{P(X \text{ and } Y)}{P(Y)}.$$

Solving each of the previous two equations for P(X and Y) shows that

$$P(Y|X)P(X) = P(X|Y)P(Y)$$

or

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}.$$

Bayes' Theorem expresses one conditional probability in terms of the other. It captures how a probability should change to account for related evidence.

Example: Suppose 231 college teens have a parent with a college degree; 214 college teens have no parent with a college degree; 49 teens who graduated from high school and did not go to college have a parent who went to college; and 298 non-college teens have no parent with a college degree. (This problem is adapted from OpenIntoStatistics, David M. Diez, Christoper D. Barr, and Mine Cetinkaya-Rundel, Creative Commons, 2016.)

Do the following problems and interpret your answers in complete sentences.

(a) Calculate the probability that a teen drawn from this pool goes to college, given that they have a parent with a college degree.

PC: A parent of the teen has a college degree

PN: Neither parent of the teen has a college degree

C: The teen is in college

NC: The teen graduated from high school and is not in college

There are 792 teens

(a) P(C | PC) = P(C and PC)/P(PC)=231/280 #The probability that the teen goes to college, given that they have a parent with a college degree is .825.

(b) Calculate the probability that a teen drawn from this pool does not go to college, given that they have a parent with a college degree.

P(NC | PC) = P(NC and PC)/P(PC)=49/280 The probability that a teen does not go to college, given that they have a parent with a college degree is:

(c) Calculate the probability that a teen drawn from this pool has a parent with a college degree, given that the teen is in college.

P (PC | C) = P(PC and C)/P(C)= 231/792 / ((231 + 214)/792) The probability that the teen has a parent with a college degree, given that the teen is in college is approximately .519.

(d) Verify that Bayes' Theorem holds for some data from this problem.

According to Bayes' Theorem, P(C | PC) = P(PC | C) P(C) / P(PC) (which agrees with (a)).

(e) Which of the above quantities do you consider to be of the most practical interest?

I consider (a) the probability that a teen goes to college, given that they have a parent with a college degree, to be of the most practical interest because it is interesting to consider whether the teen's presence in college is related to whether the teen has a parent with a college degree.

(f) Suppose we are trying to make decisions about whether a teen who graduates from high school is likely to go promptly into college. Calculate the most relevant probabilities, explain their significance in the context of the practical problem at hand, and describe possible limitations of their applicability.

To look at the likelihood that a teen goes to college, we can look at whether they have a parent with a college degree.

Part (a) shows that the probability that the teen goes to college, given that they have a parent with a college degree is 82.5%.

(Part (b) shows that the probability that the teen goes to college, given that they do not have a parent with a college degree is 17.5%.)

This result seems to suggest that if a parent has a college degree, the teen is likely to go into college. One limitation of this conclusion is that it is based only on the data provided. We would need reasons to believe that the data are representative of all teens who graduated from high school, both those in college and those not in college.

Another thing to be cautious about is whether a parent's education is relevant to whether the teen goes to college promptly. For example, if instead of checking parent's education we checked whether the parent's favorite color is purple, we might see that 82.5% of teens who have a parent with the favorite color purple are in college and 17.5% of the teens who have a parent with the favorite color purple are not in college, and this would simply reflect that going to college is more common than not, rather than indicate that parental preference for purple is relevant.

5.4.1 Exercises

Exercises

Exercise 1: Suppose 35 spam email messages contain the word "free"; 25 spam messages do not contain the word "free"; three of non-spam messages contain the word "free"; and 37 non-spam messages do not contain the word "free". (This problem is adapted from OpenIntoStatistics, David M. Diez, Christoper D. Barr, and Mine Cetinkaya-Rundel, Creative Commons, 2016.)

Do the following problems and interpret your answer in complete sentences.

(a) Calculate the probability that a message is spam, given that it contains the word "free". (b) Calculate the probability that a message is not spam, given that it contains the word "free". (c) Calculate the probability that a message contains the word "free", given that it is spam. (d) Verify that Bayes' Theorem holds for some data from this problem. (e) Which of the above quantities do you consider to be of the most practical interest? (f) Suppose we are trying to make decisions about whether a message is spam. Calculate the most relevant probabilities, explain their significance in the context of the practical problem at hand, and describe possible limitations of their applicability.

Exercise 2: Suppose the disease meningitis causes a stiff neck 70% of the time, the prior probability of any patient having meningitis is 1/50,000, and the prior probability of any patient having a stiff neck is 1/100. (Artificial Intelligence: A Modern Approach (Third Edition), Stuart J. Russell and Peter Norvig, Prentice Hall, Upper Saddle River, 2010). Use Bayes' Theorem to calculate the probability that a patient has meningitis, given they have a stiff neck. Show how you apply Bayes' Theorem. Interpret the results in a complete sentence.

```
import math
import numpy as np
import matplotlib.pyplot as plt
import random
random.seed()
```

5.5 Probability Distributions

A probability distribution is a mathematical function that maps each possible numerical outcome X of an experiment to its probability. Probability distributions come in many different types. Some fundamental ones include uniform, normal and binormal.

5.5.1 Uniform distribution

The graph of a **uniform distribution** could look like this:

In the uniform distribution, each value along the horizontal axis is equally probable.

Mathematical distributions are important. We use them to determine probabilities for results of certain random experiments and to approximate real frequency distributions.

For example, if we use Python to generate a random integer between 1 and 6, the number is selected from a uniform distribution: Each outcome is equally likely. We see a possible output below.

```
print(random.randint(1, 6))
```

```
6
```

A histogram is a visualization of a frequency distribution: It shows how often a random variable takes on a value. It shows the shape, center, and spread.

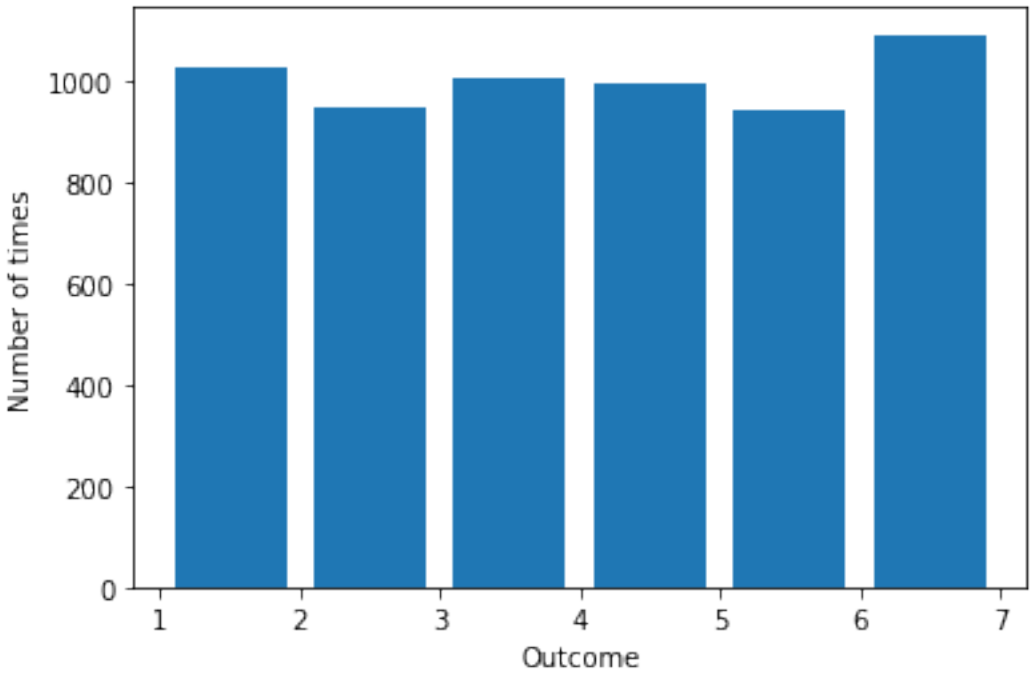
For example, consider the simulation of rolling a die 6000 times. We can capture the results with a histogram in Python using the **matplotlib.pyplot** package. `Matplotlib` contains a variety of functions to create visualizations of data. It captures the functionality of the proprietary `Matlab`.

In this section we use the `Numpy package`. It provides tools for numerical computing, e.g., efficient numerical vector/array calculations based on underlying libraries in the Fortran and C programming languages.

Here we call the `hist` method on the data values that fall in the range 1 to 7 inclusive, where each bar has width 0.8 using . Experiment to see how this call is an improvement over `plt.hist(values)` and `plt.hist(values,`

`bins = 6)` and `plt.hist(values, bins = 6, rwidth = 0.8)`. We improve the layout of the bars by specifying the leftmost and rightmost values.

```
values = [] # Work with list or array.
for i in range(6000): # Roll the die 6000 times.
    roll = random.randint(1, 6)
    values.append(roll)
# Plot a histogram of the values.
plt.hist(values, bins = range(1, 8), rwidth = 0.8)
# The leftmost edge is 1, and the rightmost is 7.
plt.xlabel('Outcome')
plt.ylabel('Number of times')
plt.show()
```



The results of the simulation of 6000 die rolls look approximately uniform with some slight differences in the frequencies of the rolls of the six numbers.

The uniform distribution approximates real data, like births by month, as shown in the figure. The real data for the years shown in the figure have some slight month-to-month differences.

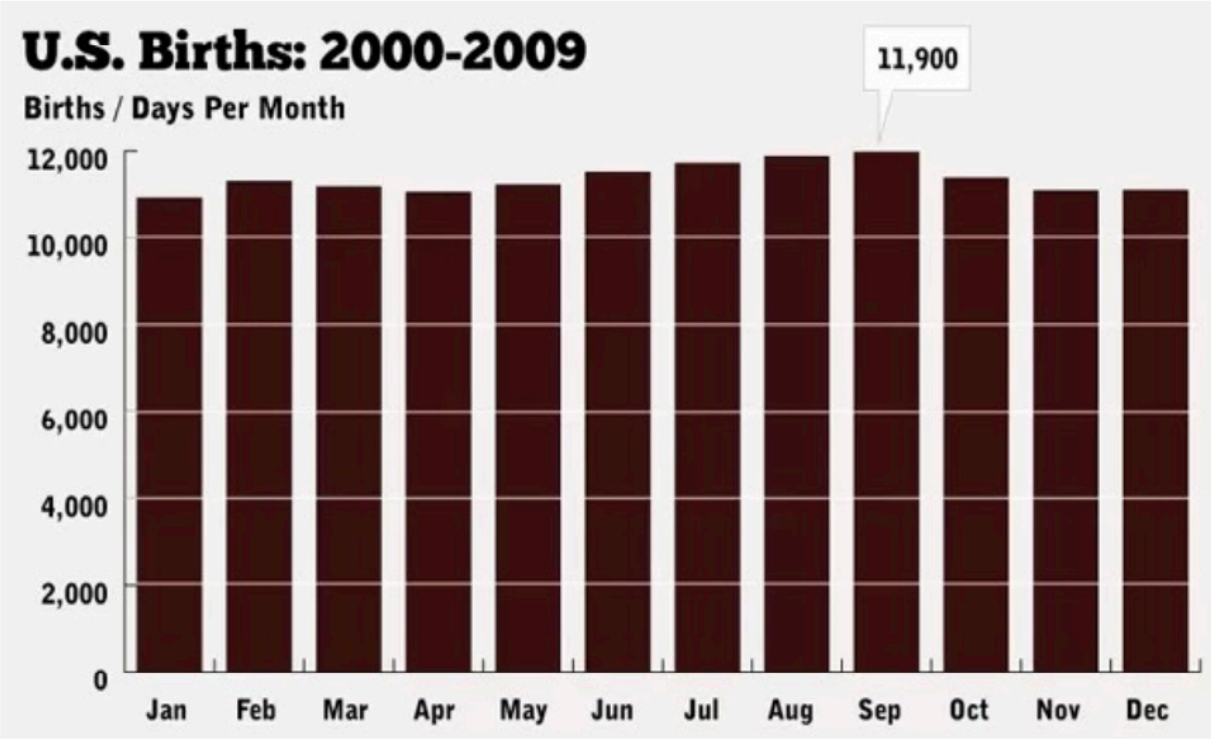
5.5.2 Normal distribution

A probability distribution can be discrete or continuous, depending on whether it maps a discrete value or a continuous value X of an experiment to its probability. A numerical variable is discrete if there are jumps between the possible values. A numerical variable is continuous if it can take on any real-number value between two other values.

The data in the previous examples are discrete. There is a jump of 1 between each possible value on the die. There is a jump of 1 between each possible number of births per month. We cannot roll a 3.72 on a die or have 10,007.21 births per month.

The graph of a **normal distribution** or Gaussian distribution could look like this:

In the normal distribution, the values in the middle ranging along the horizontal axis are more probable than at the ends ("in the tails") of the distribution. The normal distribution is often called a "bell curve" because it is shaped like a bell.


U.S. Births: 2000-2009
Births / Days Per Month
11,900
12,000
10,000
8,000
6,000
4,000
2,000
0
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

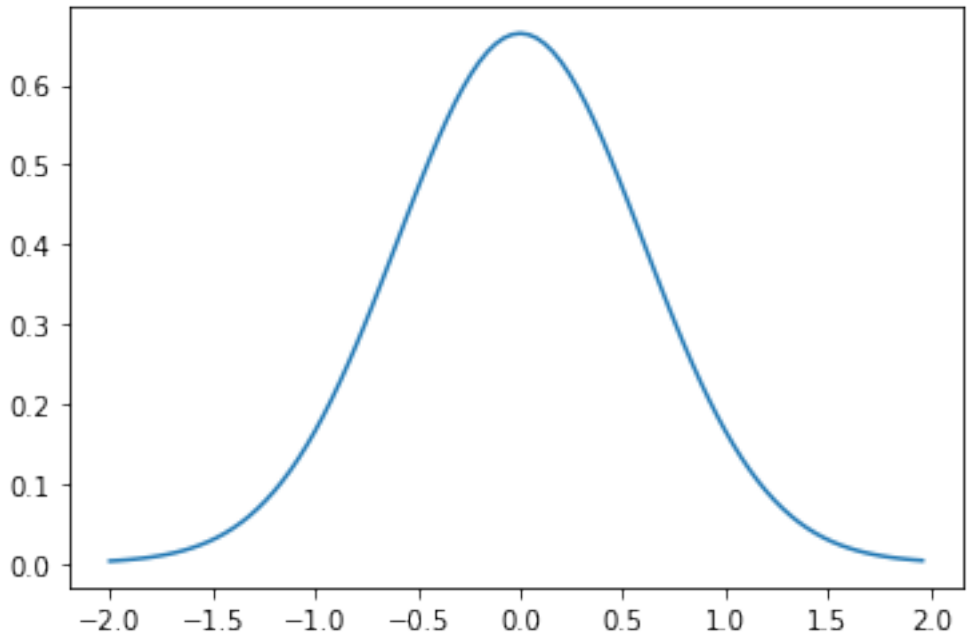
Normal
(Gaussian)

The distribution could be discrete. The continuous form of the normal distribution

$$P(X) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{X-\mu}{\sigma}\right)^2\right),$$

where μ is the mean, and σ is the standard deviation, can be graphed in Python as:

```
def plot_normal_curve(start, end, mu, sigma):
    # Plot the normal distribution with mean mu and standard deviation sigma on an␣
↪interval from start to end.
    y_coords = []
    x_coords = np.arange(start, end, (end-start)/100) # Use 100 points between start␣
↪and end.
    for x in x_coords:
        y_coords.append(1/sigma/math.sqrt(2*math.pi)*math.exp(-1/2 * ((x-mu)/
↪sigma)**2))
    plt.plot(x_coords, y_coords)

plot_normal_curve(-2, 2, 0, 0.6)
```



Distributions, such as the normal distribution, are useful for determining probabilities for results of certain random experiments, e.g., in mathematical statistics. **Statistics** is the science of collecting and analyzing data in large quantities, especially for making inferences about a **whole population** from a **representative sample**.

For example, how can we estimate the average height of all women in the country? Here the population is women in the country. The average height is the **expected value** or population parameter.

1. Sample some women at random. Measure them and average their heights to get the sample average or **sample mean**.
2. Repeat the process of sampling, measuring, and averaging.
3. Average the average heights for the samples.

The Law of Large Numbers could be casually stated as: The sample averages converge to the expected value, as the number of samples is increased.

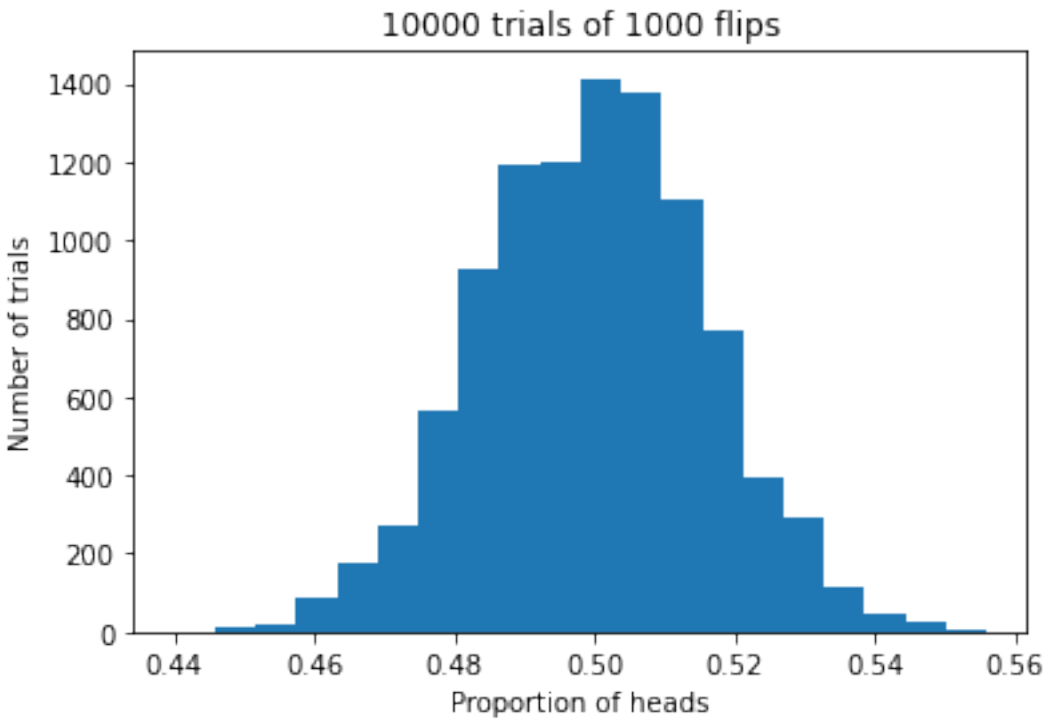
The Central Limit Theorem could be casually stated as: The distribution of the sample statistic is approximately normal, if the sample size is large enough.

A sample statistic could be a sample mean for a **numerical variable** (like the average height of a sample of people) or a **sample proportion** for a **categorical variable** (like the proportion of adults in a sample of people who are in the "yes" category for the variable that measures completion of a sixth-grade education). Note that for **numerical variables**, quantities like the average, sum, or difference of their values have clear meaning. The possible values of **categorical variables** are called "levels", such as the yes/no "levels".

Experiment with / illustrate the law and theorem using `Python`.

```
def flip(num_flips):
    head_count = 0
    for i in range(num_flips):
        if random.choice(('H', 'T')) == 'H':
            head_count += 1
    return head_count/num_flips

def flip_trials(num_flips_per_trial, num_trials):
    # Return the proportions of heads in num_trials trials of num_flips coin flips.
    # Return the mean of the proportions.
    fraction_heads = []
    for i in range(num_trials):
        fraction_heads.append(flip(num_flips_per_trial))
    mean = sum(fraction_heads) / len(fraction_heads)
    return fraction_heads, mean

def make_histogram_of_proportions(num_flips, num_trials):
    # Make a histogram of the proportions of heads in num_trials trials of num_flips␣
↪coin flips.
    plt.title(f'{num_trials} trials of {num_flips} flips')
    plt.xlabel('Proportion of heads')
    plt.ylabel('Number of trials')
    proportions_heads, mean_of_proportions = flip_trials(num_flips, num_trials)
    plt.hist(proportions_heads, bins = 20)
    xmin, xmax = 0.4, 0.6
    plt.figure()
    print(f'The mean of the proportions is {mean_of_proportions}.')

make_histogram_of_proportions(1000, 10_000)
```

```
The mean of the proportions is 0.49986099999999817.
```




```
<Figure size 432x288 with 0 Axes>
```

5.5.3 Binomial distribution

Consider a fixed number n of independent experiments, where each experiment can result in "success" with the same probability p. The binomial distribution maps the non-negative integer k to the probability of achieving exactly k successes in n experiments.

The graph of a **binomial distribution** could look like this:

The distribution is discrete. The binomial distribution

$$P(k) = \frac{n!}{k!(n-k)!}\, p^k (1-p)^{n-k}$$

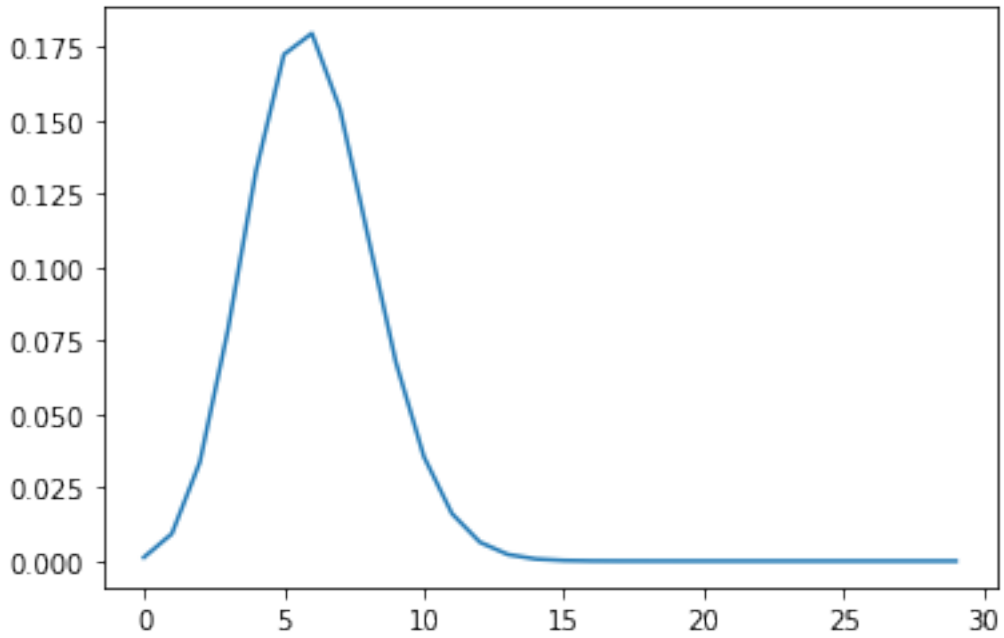
can be graphed in Python as:

```
def plot_binomial_curve(n, p):
    # Plot the binomial distribution for n independent trials, each with probability␣
↪p of success.
    y_coords = []
    for k in range(n):
        y_coords.append(math.factorial(n)/math.factorial(k)/math.factorial(n-k) *␣
```

(continues on next page)

(continued from previous page)

```
    ↪p**k * (1-p)**(n-k))
        plt.plot(np.arange(0, n, 1), y_coords) # Plot x-coordinates 0, 1, 2, 3, ..., n␣
    ↪against y-coordinates.

plot_binomial_curve(30, 0.2)
```



5.5.4 Exercises

Exercises

Exercise 1: Consider points (x, y) in a 1 x 1 square in the first quadrant of the plane $(x, y) \in [0, 1) \times [0, 1)$. Simulate filling the square with dots by picking points in the square at random. Generate the coordinates in each x-y pair by using Python to sample from a uniform distribution on the interval [0, 1). Use the ratio of number of dots inside a quarter circle with center at the origin to total dots to simulate the ratio of the area of the quarter circle to the area of the square. Use this ratio to to estimate the value of π.

Exercise 2: Use Python to simulate a game board in which 10,000 balls are dropped down onto a triangle of ten levels of pegs, one-by-one. Level 1 has 1 peg. Level 2 below it has 2 pegs, etc., down to 10 pegs at Level 10 at the bottom. Assume each ball has equal probability of moving to the left or to the right at each level.

(a) Plot the number of balls against the landing-peg number. Include a descriptive title and labels for both axes on your plot.

(b) How does the graph tend to change if you use more levels and keep the number of balls the same? Why?

(c) How does the graph tend to change if you use more balls and keep the number of levels the same? Why?

Hint: To keep a record of how many balls land in each position, start with a list or array total_balls, consisting of ten zeroes. For each of 10,000 trials (each drop of a ball), start a current_position variable for the ball at 0. Loop through levels 1 to 9. (Start at Level 1 and go up to and not including the bottom row.) Increment current_position by random.randrange(2) at each level to represent a drop to the next level, where, say, 0 represents a move left, and 1

represents a move right. When you're done with all levels, increment the ball total at the index corresponding to the position where the ball ended up using, say, total_balls[current_position] += 1.

Exercise 3: Consider a biased coin that lands on heads 20% of the time. Simulate 30 flips of the coin. Record whether no heads were flipped, 1 head was flipped, 2 heads were flipped, 3 heads were flipped, etc., up to 30 heads. Repeat the experiment 10,000 times. Draw a histogram that shows how often each number 1-30 of heads was flipped. Describe the shape and spread of the graph. Compare it to the graph of the mathematical distribution given by the formula.

Hint: Note that one way to simulate the flip of a fair coin, is as shown in the cell below because a random number generated in the left half of the interval $[0, 1)$ represents a flip of heads, and a random number generated in the right half of the interval represents tails:

With a different value of p than 0.5, you can simulate the flip of a coin biased to come up heads with a probability of p. You can increment a variable heads_counter (initialized to 0) with each simulated flip of the coin that comes up heads to check that your simulation performs as intended.

```
p = 0.5
if random.uniform(0, 1) < p:
    print("You flipped heads!")
else:
    print("You flipped tails!")
```

```
You flipped tails!
```

```
import math
import numpy as np
import matplotlib.pyplot as plt
import random
random.seed()
```

5.6 Random Walks

A **random walk on the number line** is a path in which each of n steps is taken either to the left or right at random. In an **unbiased** random walk, each step has a 50-50 chance of going to the left or to the right.

A **random walk on a square lattice** is a path in which each of n steps is taken left, right, up, or down at random. In one example of a biased random walk, a person takes a step left with probability 0.35, a step right with probability 0.2, a step forward with probability 0.4, and a step back with probability 0.05. Note the probabilities add to 1, meaning there are no other options at each iteration.

A random walk is an example of a **stochastic process** because it yields varied output due to randomness when it is executed repeatedly with the same input. By contrast, a deterministic process yields the same output when it is executed repeatedly with the same input. Random walks can be used to model (describe) practical processes in a variety of areas of study, such as the motion of particles in physics, changes in DNA and "genetic drift" in biology, the changes in stock prices in financial markets, and human migration and relocation in global geography and societies.

In a traditional framing of a random walk, a drunk person walks on a street with an equal probability of movement to the left or right at each pace. One can readily ask many questions about the random walk that can be surprisingly challenging to answer via basic calculation and pencil-and-paper experimentation.

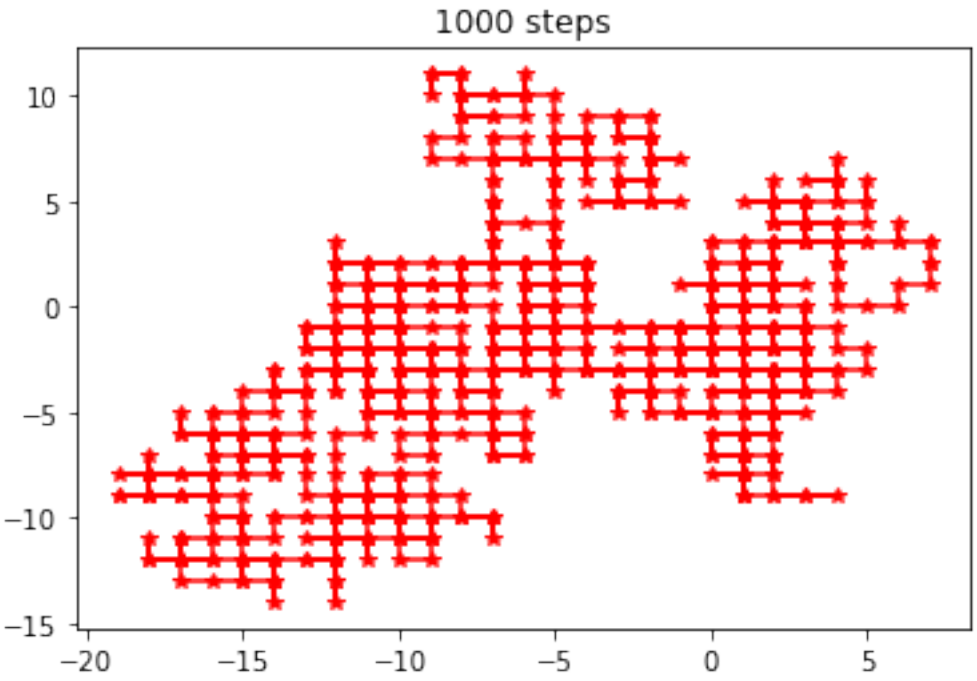
For instance, in an unbiased random walk on the number line, what is the expected number of paces the person should take in order to move one pace to the left of the starting point? Also consider, in the example above of the biased random walk on a square lattice, what is the expected number of paces the person should take in order to visit the location one pace to the left of the starting point?

Mathematical approaches can provide answers about random walks. However, many questions prove quite difficult to address theoretically.

Computer simulations can conveniently and intuitively provide insights into random walks. They can also illustrate the variations that occur over many trials of an experiment. In this section, we use computation to consider the questions above and more.

Example: Consider and explore this Python code for a random walk on a square lattice. We see a possible output below.

```
n = 1000
x, y = np.zeros(n), np.zeros(n)
for i in range(1, n):
    val = random.choice(['forward', 'back', 'left', 'right'])
    if val == 'forward':
        x[i] = x[i-1]
        y[i] = y[i-1] + 1
    elif val == 'back':
        x[i] = x[i-1]
        y[i] = y[i-1] - 1
    elif val == 'left':
        x[i] = x[i-1] - 1
        y[i] = y[i-1]
    else: # val == 'right'
        x[i] = x[i-1] + 1
        y[i] = y[i-1]
plt.plot(x, y, 'r*-')
plt.title(f"{n} steps")
plt.show()
print(f"Final position: {x[n-1]}, {y[n-1]}")
```

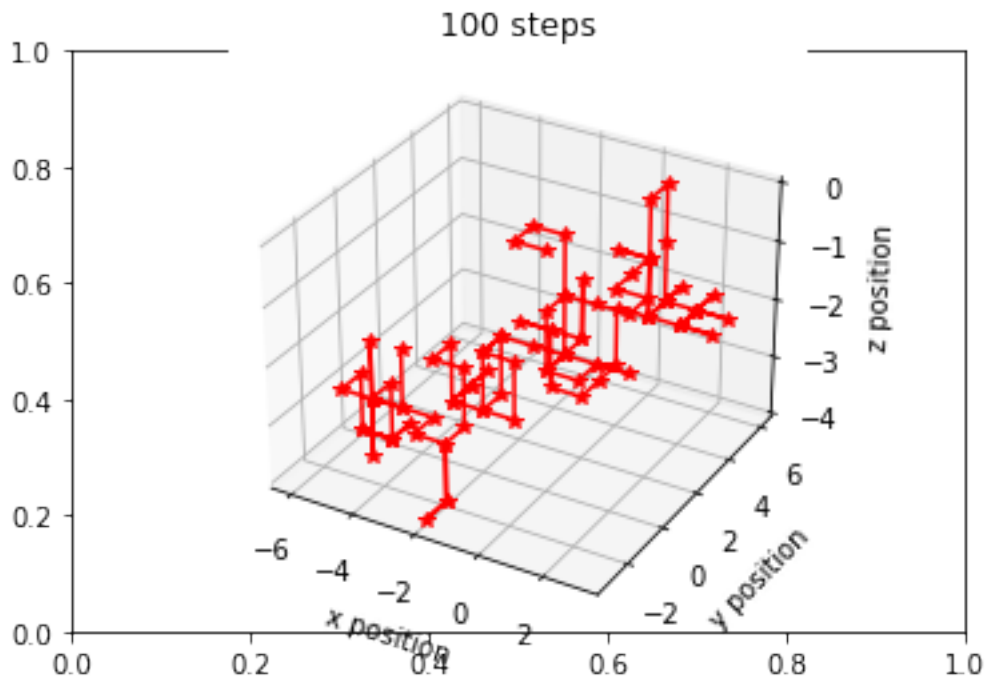



```
Final position: -10.0, 1.0
```

Example: Consider and explore this Python code for a random walk on a cubic lattice! We see a possible output below.

```
n = 100
x, y, z = np.zeros(n), np.zeros(n), np.zeros(n)
for i in range(1, n):
    val = random.choice(['forward', 'back', 'left', 'right', 'up', 'down'])
    if val == 'forward':
        x[i] = x[i-1]
        y[i] = y[i-1] + 1
        z[i] = z[i-1]
    elif val == 'back':
        x[i] = x[i-1]
        y[i] = y[i-1] - 1
        z[i] = z[i-1]
    elif val == 'left':
        x[i] = x[i-1] - 1
        y[i] = y[i-1]
        z[i] = z[i-1]
    elif val == 'right':
        x[i] = x[i-1] + 1
        y[i] = y[i-1]
        z[i] = z[i-1]
    elif val == 'up':
        x[i] = x[i-1]
        y[i] = y[i-1]
        z[i] = z[i-1] + 1
    else: # go down
        x[i] = x[i-1]
        y[i] = y[i-1]
        z[i] = z[i-1] - 1
plt.title(f"{n} steps")
ax = plt.axes(projection='3d')
ax.set_xlabel('x position')
ax.set_ylabel('y position')
ax.set_zlabel('z position')
plt.plot(x, y, z, 'r*-')
print(f"Final position: {x[n-1]}, {y[n-1]}, {z[n-1]}")
```

```
Final position: -4.0, -1.0, -2.0
```



Note you can change the figure size, for example by inserting the following line before plotting:

```
fig = plt.figure(figsize = (10, 10), dpi = 50)
```

```
<Figure size 500x500 with 0 Axes>
```

5.6.1 Exercises

Exercises

Exercise 1: A person walks on a street with an equal probability of movement to the left or right at each pace. Write a Python program to approximate the expected number of paces the person should take in order to move one pace to the left of the starting point. Suggestion: Use 0 as the starting point. State your conclusions in a complete sentence.

Exercise 2: Consider an unbiased random walk on a square lattice. Write a Python program to find approximate answers to the following questions:

(a) What is the expected number of a paces the person should take in order to visit the location one pace to the left of the starting point (0, 0)? Explain.

(b) How far away from the starting point (0,0) does a person end up after n steps of an unbiased random walk on a square lattice? Explain.

(c) If you run the simulation program for Part (b) many times, is there any statistical distribution you can expect? Explain.

Exercise 3: A person walks one pace left with a probability of 0.35, one pace right with a probability of 0.2, one pace forward with a probability of 0.4 and one pace backward with a probability of 0.05. Write a Python program to approximate the expected number of paces the person should take in order to visit the location one pace to the left of the starting point. State your conclusions in a complete sentence.

Exercise 4: A person walks one pace left, right, forward or back with equal probability. When the person walks left, right, or back, she walks one pace. When she walks forward, she walks two paces. Write a Python program to approximate how far forward the person has walked in 1000 paces. State your conclusions in a complete sentence.

Exercise 5: A **Markov chain** describes a sequence of possible events in which the probability of each event depends on the state in the previous event. For example, suppose that we modify an unbiased random walk on a square lattice such that if the previous step was to the right, then the probability of making the current move up, down, left, or right is 0.2, 0.2, 0.4, or 0.2, respectively. As such,

(a) explain the implications of a move to the right in practical terms in a complete sentence.

(b) run some simulations and address the question: How does the use of the Markov chain change the unbiased random walk on a square lattice?

Exercise 6: Two people get lost in a shopping mall. They both want to find each other as soon as possible. However, each has no clue where the other one is. There are two strategies available:

(1) One person stays in the same place. The other moves around.

(2) Both people move.

Which strategy is better for them to meet?

Do simulations and explain how they support your answer.

```
import random
```

5.7 Agent-based Models

Agent-based models are used to describe actions and interactions of individuals (agents), which occur with various likelihoods. Simulations on such models can be used to make predictions about the future state of the agents. In this section, we refer to agents acting independently. We do not address multi-agent systems in which individuals collaborate with one another in groups.

Like random walks, agent-based models can be used to capture practical processes probabilistically in a variety of areas of study. They can predict the spread of disease, forecast the segregation (separation) of different populations in a certain region, optimize traffic conditions, and describe behaviors within social networks.

In a classic example, consider that we often hear news from other people we know. If a piece of information is shared, how might it spread, from one agent to another?

Let us develop an agent-based model and use the model to simulate how a piece of information spreads over time. As with any model, we make a few underlying assumptions. Some are very practical. Some are explicit simplifications of a real-life scenario to make the problem easier to solve. In modeling and simulations, practitioners like to **start with simple models** and evaluate how well they capture real scenarios. Modelers address some shortcomings in the simulations with changes to the model assumptions. Over time, we typically expand the scope of the model to address additional scenarios. Here we assume:

1. The population is a closed community. The number of individuals does not change.
2. Agents who do not have the information do not pass any information to others!
3. An agent does not pass information to itself.
4. Once an agent hears the piece of information, it will not forget the information. There is no transition from "knowing" to "not knowing".
5. A transmission rate defines how many others hear the information from a given individual in one day.

6. Each agent has one of the allowed transmission rates with equal probability.
7. When an agent shares the information with another individual, that individual is selected from all the other members of the population with equal probability.

In models, **initial conditions** provide "starting information". Let us take the following initial condition for our problem: On Day 0, 1 individual has the information.

Parameters are changeable values that quantify practical aspects of entities or processes involved in the problem. Let us identify some parameters for the agent-based model and assign them values as follows:

1. The value of the population parameter is 1000. (There are 1000 agents.)
2. The allowed transmission rates are 0, 1, 2, 3, 4, and 5.

The way the information could spread is as follows: We have Persons 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,…, 1000. These people might have transmission rates 3, 3, 2, 0, 5, 1, 1, 2, 3, 4, 2, 1, …, 4 respectively. Initially, 1 person knows the information. If that person is Person 1, and Person 1 has a transmission rate of 3, then on the first day of gossip, Person 1 might spread the information to Persons 5, 8, and 9 with transmission rates 5, 2, and 3, respectively. In Day 2 of gossip, Person 1 spreads the information to 3 people, Person 5 to 5 people, Person 8 to 2 people and Person 9 to 3 people. Note these may be some of the same people or people who already know the information. The process continues day-by-day.

Example. Simulate the spread of information under the given assumptions. How many days does it seem to take in general for the information to spread to a quarter of the population?

```
def rumor_spread(total_population):
    # Set a maximum number of iterations.
    max_it = 1000
    # Inform Agent 0.
    informed_status = [1]
    # No one else has been informed.
    for i in range(total_population-1):
        informed_status.append(0)
    # Generate a random number from 0 to 5 for each agent and store in a list as the␣
↪number of transmissions.
    transmission_numbers = []
    for i in range(total_population):
        transmission_numbers.append(random.randint(0, 5))
    # Simulation of the process
    day = 0
    spread_to_quarter = False
    day_spread_to_quarter = 0
    while informed_status.count(1) != total_population and (day < max_it):
        # Record the first day that at least 1/4 of the population has the␣
↪information.
        if not(spread_to_quarter) and informed_status.count(1) >= int(total_
↪population/4):
            day_spread_to_quarter = day
            spread_to_quarter = True
        new_status = informed_status[:]
        for agent_index in range(total_population):
            # print(informed_status)
            if informed_status[agent_index] == 1:  # If agent is informed,
                # transmit to (inform) the appropriate number of randomly selected␣
↪agents (excluding self).
                for informing in range(transmission_numbers[agent_index]):
                    recipient_index = random.randrange(total_population)
                    while recipient_index == agent_index:
                        recipient_index = random.randrange(total_population) # Ensure␣
↪agent does not transmit to self.
```

(continues on next page)

(continued from previous page)

```
                    new_status[recipient_index] = 1
        # Copy the lists to ensure that informed_status and new_status point to two␣
↪different lists in memory.
        informed_status = new_status[:]
        day += 1
    if day_spread_to_quarter > 0:
        print(f"(A quarter of the individuals are informed by Day {day_spread_to_
↪quarter}.)")

rumor_spread(1000)
```

```
(A quarter of the individuals are informed by Day 5.)
```

Run the code repeatedly: Typically it seems to take about 5 days for the information to spread to a quarter of the population of 1000. Sometimes the code does not give any output. Why?

To answer this question, notice that the outer while loop counts up the days and continues to repeat while not everyone in the population has been informed and the maximum number of iterations has not been exceeded.

In the exercises, you will add a few lines to the code to print how many days it took for all the individuals to be informed. If not all the individuals were informed within the maximum number of iterations, then your modified code will print a message accordingly.

5.7.1 Exercises

Exercises

Exercise 1: Simulate the spread of information under the given assumptions. (a) How many days does it seem to take in general for the information to spread to half of the population? Modify the code given in this section to answer this question. (b) Add an if-else statement to the end of your function such that if the number of informed individuals equals the total population (if informed_status.count(1) == total_population), a message is printed that says by what day all the individuals are informed, i.e., include the value of day in the output. Otherwise (if the number of informed individuals does not equal the total population), print a message that not all the individuals were informed within the maximum number of iterations.

Exercise 2: This agent-based model could also be applied to the spread of disease. If "knowing the information" means "having the disease" and "not having the information" means "not having the disease", what are the interpretations of the assumptions, initial condition, and parameters that are given? Explain how you may find the model compelling in the context of disease, or what changes to the model you recommend to better reflect the spread of disease.

5.8 Mathematical Games

Many games that people play for pleasure have mathematical aspects. Indeed **game theory** is a field of mathematics.

We can identify properties of games mathematically and via computer simulation. In this section, we focus on modeling, simulation, and visualization of the results of a few games that involve randomness. The goal is to get insights into the likelihoods of different outcomes, given rules, rather than actually to engage in games for pure entertainment. No prior familiarity with the games is assumed, and we do not include any betting, which some people consider morally objectionable.

5.8.1 Craps

The game of craps uses two six-sided dice. The rules for two versions of the game are as follows:

1. "Pass Line":

- If the first roll is a 7 or 11, you win.
- Otherwise, if the first roll is a 2, 3, or 12, you lose.
- In all other cases, the specific number rolled is "the point", and you roll again until one of the following outcomes:
  - If you roll "the point" again, you win.
  - If you roll a 7, you lose.

2. "Don't Pass":

- If the first roll is a 7 or 11, you lose.
- Otherwise, if the first roll is a 2, 3, or 12, you win.
- In all other cases, the specific number rolled is "the point", and you roll again until one of the following outcomes:
  - If you roll "the point" again, you lose.
  - If you roll a 7, you win.

If you neither win, nor lose, we call the result a draw or a tie.

We can implement these two versions of the game in Python. We see possible output below.

```
def roll_two_dice():
    die1 = random.randint(1, 6)
    die2 = random.randint(1, 6)
    return die1 + die2

def pass_line():
    roll = roll_two_dice()
    if (roll == 7) or (roll == 11):
        return "win"
    elif (roll == 2) or (roll == 3) or (roll == 12):
        return "lose"
    else:
        the_point = roll
        roll = roll_two_dice()
        if (roll == the_point):
            return "win"
        elif (roll == 7):
            return "lose"
        else:
            return "draw"

def dont_pass():
    roll = roll_two_dice()
    if (roll == 7) or (roll == 11):
        return "lose"
    elif (roll == 2) or (roll == 3) or (roll == 12):
        return "win"
    else:
        the_point = roll
        roll = roll_two_dice()
        if (roll == the_point):
```

(continues on next page)

(continued from previous page)

```
            return "lose"
        elif (roll == 7):
            return "win"
        else:
            return "draw"

print("Pass line:")
print(pass_line())
print("\nPlay again\n")
print("Don't pass:")
print(dont_pass())
```

```
Pass line:
draw

Play again

Don't pass:
lose
```

Explore these two versions of the game craps in the Exercises section.

5.8.2 Choosing the Right One

Consider the following problem:

You arrive at a hotel and are invited to walk down the hall and choose what you consider the best out of n rooms. The rooms are varied with no underlying organization. You will view them one-by-one, and the person in charge says you must accept or reject a room immediately after viewing it because a stream of others seeking lodging is coming in behind you. If you have not accepted a room before you reach the end of the hall, you will get the last of the n rooms.

Problem: How can you maximize the chance that you will accept the room you consider the best?

Optimal stopping problems like this one arise in various contexts, including financial markets (choosing when to sell stock in an n-day period), housing markets (choosing when to sell a house in an n-day period), and hiring or marriage (selecting someone when n candidates "dribble in" sequentially).

Notice that if your strategy is to pick a room at random, your chance of getting the best room is $1/n$. If your strategy is to choose the first, second, last, or "middle" room, your chance of getting the best room is also $1/n$. For larger values of n, this strategy performs increasingly poorly.

An insight here is that you can benefit from a "preview process." Set a strategy in which you preview r rooms (rejecting each one) and record the maximum score m from the preview set. Then pick the first room with a score that is better than m.

For example, suppose $n = 10$. Previewing r rooms is possible if $0 \leq r < n$. Suppose your strategy is to preview $r = 3$ rooms. If the scores for the rooms are [67, 91, 85, 92, 2, 71, 32, 56, 37, 34], then $m = 91$. The first room with a score better than m is the fourth room with a score of 92. In this case, the strategy yielded the best room.

We can explore the game of choosing the right one with the following algorithm:

```
# 1. Generate a list ratings of n rooms, from 0 to 100.

# 2. Define the number_rooms_to_preview as int(0.37 * n).

# 3. Find max among the first 37% by looping through the rooms to be visited.
```

(continues on next page)

(continued from previous page)

```
# 4. While rooms still remain to be visited,
# and the rating does not surpass the first 37% of ratings,
# advance to the next room:
# while (i < n) and (ratings[i] <= max):
    # i += 1

# 5. Once you exit the while loop, if you haven't reached the final room,
# if i < n:
# then you have found a room that surpasses the first 37%.
# You found the first rating that exceeds max, so assign that rating as your choice:
    # choice = ratings[i]

# 6. Otherwise, you have been through all the rooms, and you must take the last room:
# else:
    # choice = ratings[n-1]

# 7. Sort apartments best to worst:
# ratings.sort(reverse = True)

# 8. Find the index of your choice in the sorted list
# to get the rank of your chosen room (1 <= rank <= n):
# rank = ratings.index(choice) + 1

# 9. Append the rank of the selected apartment into a list.

# 10. Plot these ranks in a histogram.
```

Explore this algorithm for choosing the right one in the Exercises section.

5.8.3 Coin-flipping Game

Suppose you have a coin to flip and n tokens. In this game, you put a fraction F of your tokens (Fn tokens) on the table, $0 < F \leq 1$. You choose either heads or tails for the game, for example heads. Then you flip the coin. If it comes up with your choice (heads), then you win Fn additional tokens. If your choice is wrong (the coin comes up tails), then you lose all Fn tokens. Repeat the process, each time using the same fraction F of the number of tokens that you now have and using the same coin-flip choice (in this example, heads). Do this until you lose all your tokens or up to a maximum number of flips, for example 30, whichever comes first.

Example: You start the game with $n = 100$ tokens. You decide that you will put down $F = 1/2$ of your tokens for each turn, use heads as your guess, and play until you lose all your tokens or up to a maximum of 30 flips, whichever comes first. As such, you put 50 tokens on the table. You flip the coin. It comes up heads, so you win 50 tokens. Therefore, you have 150 tokens altogether (150 points). So, next you put 75 tokens on the table. You flip the coin. It comes up tails, so you lose the 75 tokens. Now you have 75 tokens (75 points) altogether. In one version of the game, points must be positive integers, so you chose a convention for rounding up or rounding down when the number of tokens $Fn$is odd. In another version, fractional points like 37.5 are allowed. We could simulate a continuation of the game with one of these conventions to determine how many points you have at the end of the game.

It has been mathematically established that a winning strategy to maximize your points is to put down the optimal fraction F of your tokens, given by the Kelly criterion as

$$F = \frac{bp - 1 + p}{b},$$

where p is the probability of winning, like $p = 0.5$ for heads, $1 - p$ is the probability of losing, and b are the odds in the game. For example, odds $b = 1$ means "put down 1 and win $b = 1$ points" (or lose $b = 1$ point).

Explore this game in the Exercises section.

5.8.4 Exercises

Exercises

Exercise 1: For the game of craps, does the choice of "Pass Line" vs. "Don't Pass" affect the likelihood of winning, under the rules given? If so, what kind of difference and how much difference? Use quantitative information to support your answer. State whether you have provided mathematical proof or computational evidence.

Exercise 2: If one of the two dice is biased in the game of craps, how does that affect the analysis and conclusions from Exercise 1? Illustrate your answer by doing some simulations and discussing the results.

Exercise 3: Simulate the optimal stopping problem for ten rooms using the algorithm discussed in this section. Execute many trials. (a) Explain whether you typically get the best room or not when you preview three of the rooms. (b) If you don't get the best room, how good of a room do you typically get? (c) Show a histogram of the ranks of the rooms you choose. Do the shape, center, and spread of the ranks suggest that the strategy (algorithm) is a good one for winning the best room? Explain. (d) Does the strategy of previewing three rooms tend to lead to a better outcome than if you preview two or four? Support your answer. (e) It can be shown mathematically that the optimal fraction of the total rooms that you should preview is $1/e$ as n gets very large. The probability that you will select the best room this way is also $1/e$ for large n. Are your simulations consistent with these results? Explain.

References

Dekking, F.M., Kraaikamp, C., Lopuhaa, L.E., Meester, L.E. 2005. A Modern Introduction to Probability and Statistics: Understanding why and how. The Netherlands: Delft.

Guttag, J.V. 2021. Introduction to Computation and Programming Using Python (3E) Cambridge, MA. The MIT Press.

```
import random
import matplotlib.pyplot as plt
```

5.9 Solution to Exercises

I tried to pose these questions to ask for human thought by asking the student to judge, think, etc. I intend for the student to address these questions with original thought and by referring to the chapter and any resource explicitly provided in the question, initially without doing outside research.

Many of the exercises have many answers. I have provided some possible answers from my own perspective.

5.9.1 Random Numbers

Exercise 1. Which statement do you judge to be more accurate? Why?

1. The computer program selected a number between 1 and 10 at random.
2. The computer program selected a number between 1 and 10 arbitrarily.

I consider the first statement to be more accurate because the "arbitrary" description allows for a personal preference, often unconscious, to come into play. A computer does not have a personal whim to influence picking some numbers over others, so "random" may be more accurate.

Exercise 2. When a computer program selects a number between 1 and 10, do you think that the choice involves possible preference for some number over others? Support your answer with examples.

According to this chapter, people typically speak of computers generating random numbers. (Technically the computer is doing pseudo-random number generation in which the number is produced by a step-by-step, complicated, but not random, process.) As such, I think that the process does **not** involve preference for one number over another, unless the programmer has designed the program such that one number is more likely than another to be chosen. For example, if the computer picks a (pseudo-) random number between 1 and 10 in a conventional way, we can assume each of the ten numbers is equally likely to be chosen with no preference for one number over another. However, the programmer could write a program in which 7 is chosen 50% of the time, and the other 50% of the time, 1, 2, 3, 4, 5, 6, 8, 9, or 10 is chosen with equal probability. In this case, the computer's choice could be said to involve a "preference" for 7 over the other numbers if you use "preference" to mean simply "an advantage".

Exercise 3: Think of an example of how randomness might be involved in a study investigating a new medicine. Explain how randomness is involved. Could computation, for example using Python, be helpful in the study? Explain your answer.

A study investigating a new medicine might use randomness to place each volunteer into either a group receiving the old medicine or a group receiving the new medicine. Computation could be helpful in analyzing the results because a large number of participants might be involved, and a large amount of data might be collected. Computers are good at doing a large number of detailed calculations quickly.

Exercise 4: Think of an example of how randomness might occur in the natural world. Explain how randomness is involved. Could a computer simulate or mimic a natural process in the context of your example? If so, how could a computer simulation be useful?

Molecules in the air might have slight fluctuations in temperature and position that are influenced by randomness. A computer could simulate how the particles flow and influence weather, maybe by using broader observations, rather than particle-by-particle information. This could be useful in forecasting the weather.

Exercise 5: Think of an example of how randomness might occur in a game. Explain how randomness is involved. Could a computer simulate the game? If so, how could the computer simulation be useful?

Randomness would occur in a game that involves drawing cards from a deck. A computer could simulate the game if it is programmed with the rules of the game. The computer simulation could be useful for investigating whether a certain strategy is effective by using it in many trials of the game.

Exercise 6: Consider a die (plural: "dice"). Do you think that when we roll a physical die, we can be certain that there is an equal chance of the die landing with each of the six sides facing up? Or could there be a bias toward some sides? Explain your thinking. Do you think the outcome of a roll of the physical die could be influenced by the previous roll? Explain your thinking. How could we use a computer program to simulate a die roll? What are some advantages of doing so?

I think that we cannot be certain that a physical die has an exactly equal chance of landing with each of the six sides facing up because slight imperfections in the die could produce a bias toward some sides. If the die is shaken up well before it is released, then the previous roll should not influence how the die lands. We could use a computer to simulate a die roll by having it generate a random number between 1 and 6. One advantage of using a computer simulation is that it could be used to "roll the die" many times in quick succession for purposes of testing out different ideas, such as: If you roll the die many times, you will get a 1 approximately 1/6 of the time.

Exercise 7: Consider the following answer to the question "True or false— An arbitrary choice is made for no special reason":

"Yes, an arbitrary choice typically refers to a decision or selection that is made without any specific reason or justification. It means that there is no objective or rational basis for choosing one option over another. Arbitrary choices are often made when there is no clear preference or when the decision doesn't have a significant impact on the outcome. In such cases, people may resort to random selection, personal whim, or other non-logical factors to make a choice." (ChatGPT, personal communication, June 6, 2023).

Provide a critique for which parts are right (if any). Provide justification for why they are correct. Identify which parts are wrong (if any) and explain what is wrong. Rewrite the answer in your own words with any corrections that are needed.

(Retain the citation as with any paraphrased material.) Add an example that you think of yourself to clarify the answer to the true/false question.

This answer is largely correct. When you ask a friend to pick an **arbitrary** integer between 1 and 10, you are asking them to make a selection without any specific reason or justification. In this scenario, there is only a personal or gut-level basis for choosing one option over another. We only ask a friend to pick a number arbitrarily when there is no "right answer" that they could use find by reasoning through which number to pick. The friend could pick based on personal whim or non-logical factors or even use a random number generated by a computer. The only part of the answer that I consider wrong is that the decision could have a significant impact on the outcome if you say, for example, "Pick an arbitrary integer between 1 and 10, and if you pick the number I wrote down on this card, I'll give you my lunch."

Exercise 8: Use the code below to explore Python commands in the`random`library. Read the documentation here as desired: https://docs.python.org/3/library/random.html . For each`print`statement, insert a comment (using`#`) that describes all the possible numbers that can be generated.

```
4
3
5
1
6
19.30733945140265
```

```
0
1
5
1
4
13.446055601574948
```

Exercise 9: Which does a Python program generate: a pseudo-random number or a truly random number? Give an example, and explain why the example is pseudo random or truly random.

A Python program generates pseudo-random numbers even though we call it "random number generation". For example, if we do `print(random.randrange(6))`, Python uses mathematical methods to **simulate** (mimic) randomness when it selects 0, 1, 2, 3, 4, or 5 with equal probability. The example is pseudo random because the number is produced by a step-by-step, complicated, but not random, process.

5.9.2 Probability

Exercise 1: Do Problem 5 from the beginning of the Probability section mathematically: Suppose you roll a perfect die n times and keep a record of each outcome. What is the chance or probability that the die lands on 1, 2, 3, 4, 5, and then 6 when rolled six times ($n = 6$)? Use Python to evaluate your answer. Explain your conclusion in a complete sentence.

```
(1/6)**6
```

```
2.1433470507544573e-05
```

```
2.14/100_000
```

```
2.1400000000000002e-05
```

The probability of rolling the sequence 1, 2, 3, 4, 5, and then 6 is approximately 2 in 100,000.

Exercise 2: Do Example 5 from the beginning of the Probability section computationally: Suppose you roll a perfect die n times and keep a record of each outcome. What is the chance or probability that the die lands on 1, 2, 3, 4, 5, and then 6 when rolled six times ($n = 6$)? Do a large number of trials of six rolls of a die and count the number of times the

rolls come out as 1 followed by 2, followed by 3, followed by 4, followed by 5, followed by 6. A piece of syntax you may consider using is `if (rolls[0] == 1 and rolls[1] == 2 and rolls[2] == 3 and rolls[3] == 4 and rolls[4] == 5 and rolls[5] == 6):`. Compare the fraction times you get this pattern to your mathematical answer from the previous exercise in a complete sentence.

```
2
2
0
2
1
4
4
2
3
4
2.4
```

When I use my program to do six coin flips 100,000 times, my mathematical answer says I should get the sequence 1, 2, 3, 4, 5, 6 approximately twice. Often I do get 2 "successes" when I do 100,000 trials. Sometimes I get 0 or 1 or sometimes as many as 6 or so, but typically if I do 100,000 trials ten times, the average number of successes is close to the predicted 2.14.

Exercise 3: If event A is that the die comes up 2, and event B is that the die comes up with an even number, are the two events A and B mutually exclusive? Explain. Is the addition rule $P(A \text{ or } B) = P(A) + P(B)$ applicable? If so, show that the rule holds. If not, show that the rule does not hold. Use Python to illustrate your answer.

The events are not mutually exclusive because they can happen at the same time. If the die comes up 2, then then the die comes up with an even number. Therefore, the addition rule does not apply. In this case:

```
# P(A)
pA = 1/6
# P(B)
pB = 3/6
# P(A or B) = P(B) = 1/2, not P(A) + P(B)
pA + pB
# which is approximately
```

```
0.6666666666666666
```

Exercise 4: Calculate the probability (a) mathematically and (b) computationally of obtaining the given result in n rolls of a fair six-sided die. Make a statement in a complete sentence about how many times you expect the outcome, in terms of a number of trials (consisting of n rolls per trial). Compare the fraction of times your simulation produces the target pattern with your mathematical answer.

1. $n = 6$, all numbers are the same
2. $n = 6$, at least one number is different
3. $n = 1$, the number rolled is greater than 2
4. $n = 2$, the sum of the two numbers is greater than 2
5. $n = 2$, the two numbers are different
6. $n = 3$, two numbers are the same and one is different
7. $n = 6$, all the numbers are different
8. $n = 100$, all the numbers are the same
9. $n = 6$, the number 1 appears at least once

10. $n = 6$, the number 1 appears at most once

```
# (1.)
# (a)
print("For n = 6 rolls of a fair six-sided die, the probability of getting all the␣
 ↪numbers of the same is:")
print("(1/6)**5:")
print((1/6)**5)
print("or approximately once in 10,000 trials: 1.3/10_000:")
print(1.3/10_000)
```

```
For n = 6 rolls of a fair six-sided die, the probability of getting all the␣
 ↪numbers of the same is:
(1/6)**5:
0.00012860082304526745
or approximately once in 10,000 trials: 1.3/10_000:
0.00013000000000000002
```

Exercise 4 Part 1b)

```
How many times to we get all the numbers the same in 10,000 trials when we do 10,
 ↪000 trials ten times?
2
3
2
0
1
2
1
1
2
0
Average: 1.4
```

On average we get all the numbers the same around 1 or 2 times in 10,000 trials as expected.

```
# (2.)
# (a)
print("For n = 6 rolls of a fair six-sided die, the probability of getting at least␣
 ↪one number different is")
print("1 - (1/6)**5:")
print(1 - (1/6)**5)
print("or approximately once in 10,000 trials we do NOT get at least one number␣
 ↪different.")
print("To check: Summing the answer with the previous answer gives approximately 1:")
print(1 - (1/6)**5 + 1.3/10_000)
```

```
For n = 6 rolls of a fair six-sided die, the probability of getting at least one␣
 ↪number different is
1 - (1/6)**5:
0.9998713991769548
or approximately once in 10,000 trials we do NOT get at least one number different.
To check: Summing the answer with the previous answer gives approximately 1:
1.0000013991769547
```

Exercise 4 Part 2b)

The simulation in (a) shows that on average we only get all the numbers the same around 1 or 2 times in 10,000 trials as

expected. The rest of the time we get at least one number different from what we expected.

```
# (3.)
# (a)
print("For n = 1 roll of a fair six-sided die, the number rolled is greater than 2␣
 ↪with probability:")
print("(4/6):")
print(4/6)
print("or approximately 2/3 of the trials:")
print(2/3)
```

```
For n = 1 roll of a fair six-sided die, the number rolled is greater than 2 with␣
 ↪probability:
(4/6):
0.6666666666666666
or approximately 2/3 of the trials:
0.6666666666666666
```

Exercise 4 Part 3b)

```
Fraction of times we got a roll > 2 in 100000 trials: 0.6669
```

The simulation in (b) shows we get a roll greater than 2 in about 2/3 or about 0.67 of the large number of trials as expected.

```
# (4.)
# (a)
print("For n = 2 rolls of a fair six-sided die, the number rolled is greater than 2␣
 ↪with probability:")
print("35/36")
print("or approximately 35/36 of the trials:")
print(35/36)
```

```
For n = 2 rolls of a fair six-sided die, the number rolled is greater than 2 with␣
 ↪probability:
35/36
or approximately 35/36 of the trials:
0.9722222222222222
```

Exercise 4 Part 4b)

```
Fraction of times we get two rolls with sum > 2 in 100000 trials: 0.97319
```

The simulation in (b) shows we get a sum greater than 2 in about 0.97 of the large number of trials as expected.

```
# (5.)
# (a)
print("For n = 2 rolls of a fair six-sided die, the numbers rolled are different with␣
 ↪probability:")
print("5/6")
print("or approximately 5/6 of the trials:")
print(5/6)
```

```
For n = 2 rolls of a fair six-sided die, the numbers rolled are different with␣
 ↪probability:
5/6
or approximately 5/6 of the trials:
0.8333333333333334
```

Exercise 4 Part 5b)

```
Fraction of times we get two rolls that are different from each other in 100000␣
↪trials of two rolls: 0.83389
```

The simulation in (b) shows that two rolls produce two different numbers about 0.83 of the large number of trials as expected.

```
# (6.)
# (a)
print("For n = 3 rolls of a fair six-sided die,")
print("* two numbers are the same, and one is different")
print("if")
print("* they are not all different, and they are not all the same,")
print("so we get two the same and one different with probability:")
print("1 - 1*5/6*4/6 - 1/36")
print("or approximately 1 - 20/36 - 1/36 of the trials:")
print(15/36)
```

```
For n = 3 rolls of a fair six-sided die,
* two numbers are the same, and one is different
if
* they are not all different, and they are not all the same,
so we get two the same and one different with probability:
1 - 1*5/6*4/6 - 1/36
or approximately 1 - 20/36 - 1/36 of the trials:
0.4166666666666667
```

Exercise 4 Part 6b)

```
Fraction of times that three rolls produce two numbers that are the same and one␣
↪that is different in 100000 trials of three rolls: 0.41633
```

The simulation in (b) shows that three rolls produce two of the same numbers and one different about 0.42 of the large number of trials as expected.

```
# (7.)
# (a)
print("For n = 6 rolls of a fair six-sided die, the probability that all six numbers␣
↪are different is:")
print("1 * 5/6 * 4/6 * 3/6 * 2/6 * 1/6")
print("or approximately 120/6**5 of the trials:")
print(120/6**5)
print("It happens approximately 1.5 times out of 100 on average:")
print(1.5/100)
```

```
For n = 6 rolls of a fair six-sided die, the probability that all six numbers are␣
↪different is:
1 * 5/6 * 4/6 * 3/6 * 2/6 * 1/6
or approximately 120/6**5 of the trials:
0.015432098765432098
It happens approximately 1.5 times out of 100 on average:
0.015
```

Exercise 4 Part 7b)

```
How many times to we get all the numbers different in 100 trials when we do 100␣
 ↪trials ten times?
3
3
1
0
1
1
0
1
3
2
Average: 1.5
```

On average we get all the numbers different around 1.5 times in 100 trials as expected.

```
# (8.)
# (a)
print("For n = 100 rolls of a fair six-sided die, the probability that all 100␣
 ↪numbers are the same is:")
print("1 * (1/6)**99")
print("or approximately 1/6**99 of the trials:")
print(1/6**99)
print("It happens essentially never.")
```

```
For n = 100 rolls of a fair six-sided die, the probability that all 100 numbers␣
 ↪are the same is:
1 * (1/6)**99
or approximately 1/6**99 of the trials:
9.183880244919038e-78
It happens essentially never.
```

Exercise 4 Part 8b)

```
0
0
0
0
0
0
0
0
0
Average: 0.0
```

```
# (9.)
# (a)
print("For n = 6 rolls of a fair six-sided die, the probability that the number 1␣
 ↪appears at least once is:")
print("1 - (5/6)**6")
print("or approximately 66.5% of the trials:")
print(1 - (5/6)**6)
```

```
For n = 6 rolls of a fair six-sided die, the probability that the number 1 appears␣
 ↪at least once is:
```

(continues on next page)

(continued from previous page)

```
1 - (5/6)**6
or approximately 66.5% of the trials:
0.6651020233196159
```

Exercise 4 Part 9b)

```
Fraction of times that six rolls produce at least 1 one in 100000 trials of six␣
↪rolls: 0.66659
```

The simulation in (b) shows that six rolls produce at least 1 one in about 0.67 of the large number of trials as expected.

```
# (10.)
# (a)
print("For n = 6 rolls of a fair six-sided die, the probability that the number 1␣
↪appears at most once is:")
print("(5/6)**6 + 1/6*(5/6)**5 * 6")
print("or approximately (5/6)**6 + (5/6)**5% of the trials:")
print((5/6)**6 + (5/6)**5)
```

```
For n = 6 rolls of a fair six-sided die, the probability that the number 1 appears␣
↪at most once is:
(5/6)**6 + 1/6*(5/6)**5 * 6
or approximately (5/6)**6 + (5/6)**5% of the trials:
0.7367755486968453
```

Exercise 4 Part 10b)

```
Fraction of times that six rolls produce at most 1 one in 100000 trials of six␣
↪rolls: 0.73633
```

The simulation in (b) shows that six rolls produce 0 or 1 one in about 0.7 of the large number of trials as expected.

Exercise 5: Consider the function below. Add comments to explain the code. Summarize the purpose of the function in a complete sentence. Demonstrate its performance on a sample run. Summarize the results in a complete sentence.

```
0.5027
```

The sample run does 10,000 flips. The function returns the fraction of heads, which came out to 0.4966 on one run and to 0.5027 on another.

Exercise 6: Consider the function below. Add comments to explain the code. Summarize the purpose of the function in a complete sentence. Demonstrate its performance on a sample run. Summarize the results in a complete sentence.

```
0.5029899999999993
```

The sample run does 1000 trials (in which 100 coin flips are executed and the fraction of heads is computed). The function returns the average of the 1000 fractions of heads, which came out to 0.49915999999999944 on one run and to 0.5029899999999993 on another.

Exercise 7. When a fair coin is flipped once, the theoretical probability that the outcome will be heads is equal to 1/2. Therefore, according to the LLN, the proportion of heads in a "large" number n of coin flips "should be" roughly 1⁄2, and the proportion of heads approaches 1/2 as n approaches infinity. Illustrate this idea using simulation with a larger and larger number of flips.

```
print(flip(10))
print(flip(100))
```

(continues on next page)

(continued from previous page)

```
print(flip(1000))
print(flip(10_000))
```

```
0.3
0.54
0.495
0.5039
```

When I flip 10, 100, 1000, and 10,000 times, respectively, I get fractions of heads, for example, 0.3, 0.54, 0.495, 0.5039, respectively.

5.9.3 Conditional probability

Exercise 1: Suppose 35 spam email messages contain the word “free”; 25 spam messages do not contain the word “free”; three of non-spam messages contain the word “free”; and 37 non-spam messages do not contain the word “free”. (This problem is adapted from OpenIntoStatistics, David M. Diez, Christoper D. Barr, and Mine Cetinkaya-Rundel, Creative Commons, 2016.)

Do the following problems and interpret your answer in complete sentences.

(a) Calculate the probability that a message is spam, given that it contains the word “free”.

(b) Calculate the probability that a message is not spam, given that it contains the word “free”.

(c) Calculate the probability that a message contains the word “free”, given that it is spam.

(d) Verify that Bayes’ Theorem holds for some data from this problem.

(e) Which of the above quantities do you consider to be of the most practical interest?

(f) Suppose we are trying to make decisions about whether a message is spam. Calculate the most relevant probabilities, explain their significance in the context of the practical problem at hand, and describe possible limitations of their applicability.

Exercise 1a)

```
0.9210526315789473
```

Exercise 1b)

```
0.07894736842105263
```

Exercise 1c)

```
0.5833333333333334
```

Exercise 1d)

```
# (d)
# According to Bayes' Theorem, P(S | F) = P(F | S) P(S) / P(F) or:
pFgS * 60/100 / (38/100)
# which agrees with (a).
```

```
0.9210526315789473
```

Exercise 1e)

Probably the most practical number is the probability of spam, given the email contains the word "free" because checking for spam is something you might want to do. It seems that "free" comes up a lot in spam messages. It may be a useful marker because the probability of spam given the possibly related word "free" is about 92%. One limitation of this conclusion is that it is based only on the data provided. We would need reasons to believe that the data are representative of all emails, spam and not spam, that do and do not contain the word "free". Also practically all spam messages contain the word "the", so we have to be confident that "free" is not so widespread that it isn't helpful. Here the probability that a message is not spam, given that it has the word "free" is somewhat small: about 8%.

Exercise 2: Suppose the disease meningitis causes a stiff neck 70% of the time, the prior probability of any patient having meningitis is 1/50,000, and the prior probability of any patient having a stiff neck is 1/100. (Artificial Intelligence: A Modern Approach (Third Edition), Stuart J. Russell and Peter Norvig, Prentice Hall, Upper Saddle River, 2010). Use Bayes' Theorem to calculate the probability that a patient has meningitis, given they have a stiff neck. Show how you apply Bayes' Theorem. Interpret the results in a complete sentence.

```
0.0014
```

5.9.4 Probability Distributions

Exercise 1: Consider points (x, y) in a 1 x 1 square in the first quadrant of the plane $(x, y) \in [0, 1) \times [0, 1)$. Simulate filling the square with dots by picking points in the square at random. Generate the coordinates in each x-y pair by using Python to sample from a uniform distribution on the interval [0, 1). Use the ratio of number of dots inside a quarter circle with center at the origin to total dots to simulate the ratio of the area of the quarter circle to the area of the square. Use this ratio to to estimate the value of π.

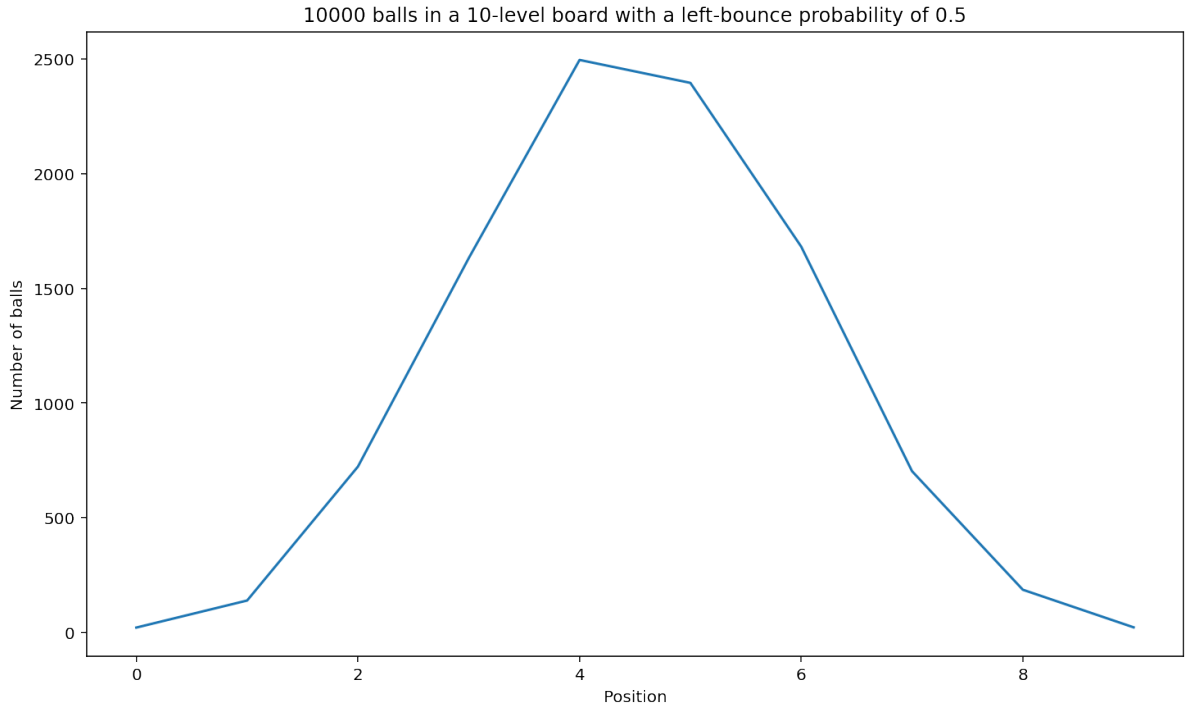
```
3.14152
```

Exercise 2: 1. Use Python to simulate a game board in which 10,000 balls are dropped down onto a triangle of ten levels of pegs, one-by-one. Level 1 has 1 peg. Level 2 below it has 2 pegs, etc., down to 10 pegs at Level 10 at the bottom. Assume each ball has equal probability of moving to the left or to the right at each level.

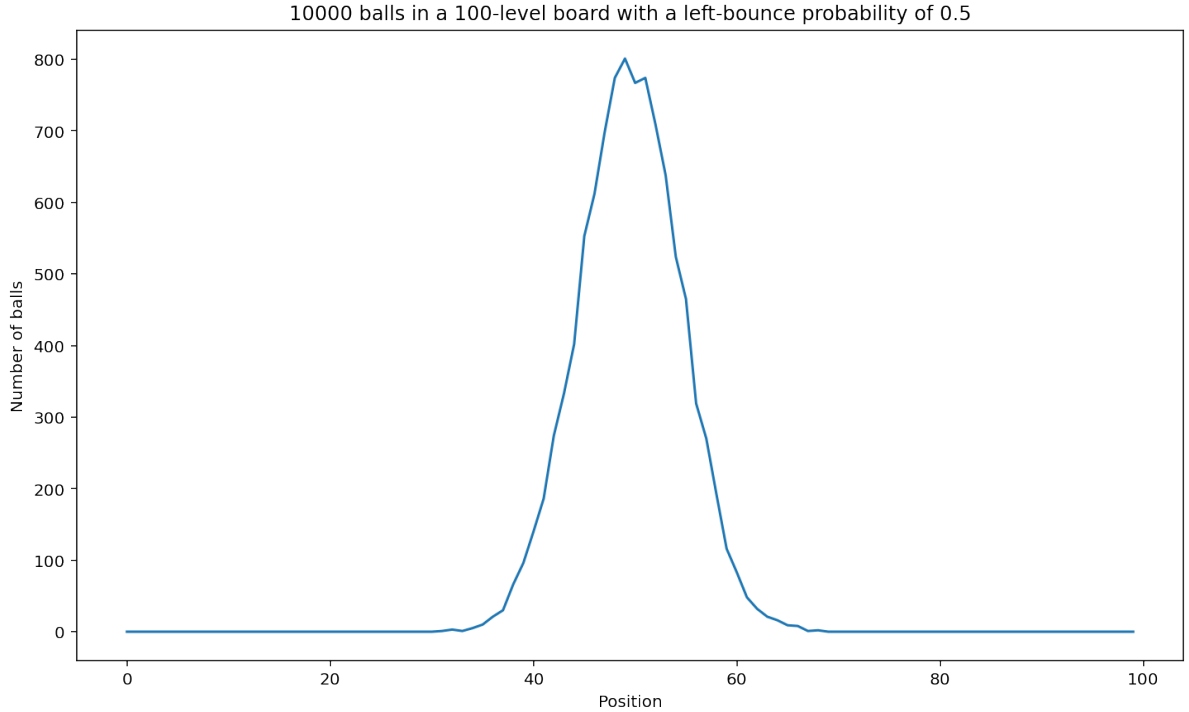
(a) Plot the number of balls against the landing-peg number. Include a descriptive title and labels for both axes on your plot.

(b) How does the graph tend to change if you use more levels and keep the number of balls the same? Why?

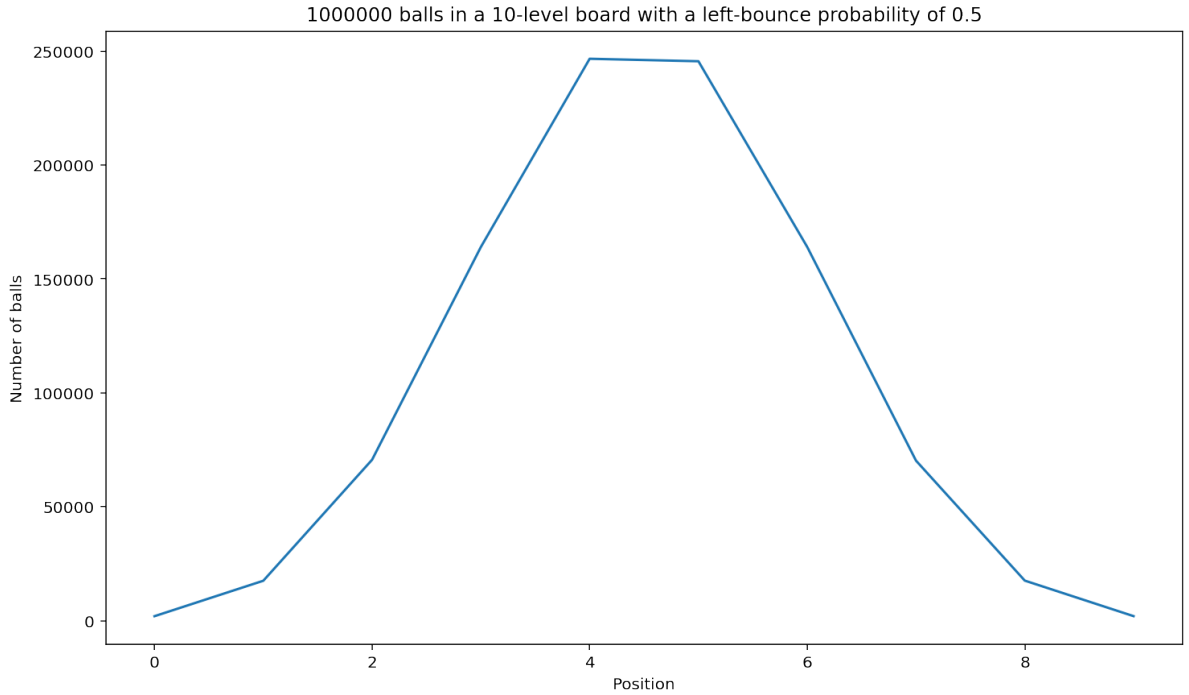
(c) How does the graph tend to change if you use more balls and keep the number of levels the same? Why?



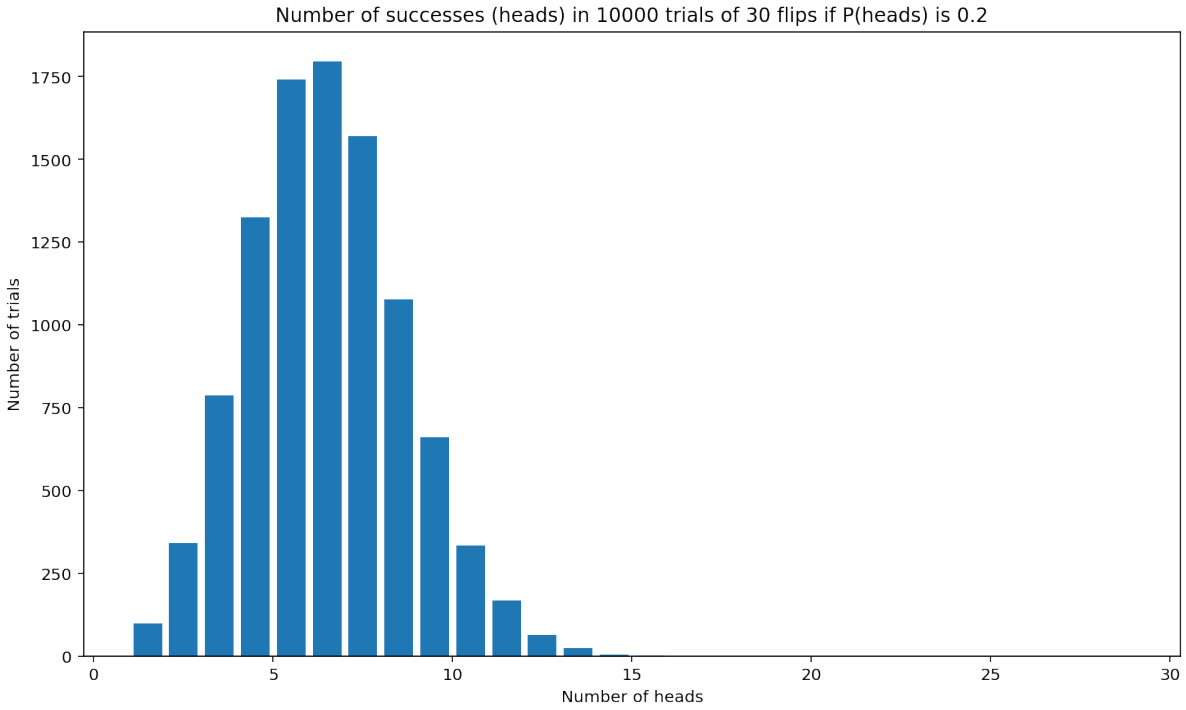

```
# If you use more levels and keep the number of balls the same, then the graph is␣
 ↪very thin in the tails because it is unlikely for a ball to bounch all the way to␣
 ↪the left or all the way to the right so many times, and the balls pile up sharply␣
 ↪in the range of "middle" positions.
board_game(10_000, 100)
```



```
# Similarly, if you increase the number of balls and keep the the number of levels at␣
 ↪10, the tails are a little chunkier than before:
board_game(1_000_000, 10)
```



Exercise 3: Consider a biased coin that lands on heads 20% of the time. Simulate 30 flips of the coin. Record whether no heads were flipped, 1 head was flipped, 2 heads were flipped, 3 heads were flipped, etc., up to 30 heads. Repeat the experiment 10,000 times. Draw a histogram that shows how often each number 1-30 of heads was flipped. Describe the shape and spread of the graph. Compare it to the graph of the mathematical distribution given by the formula.



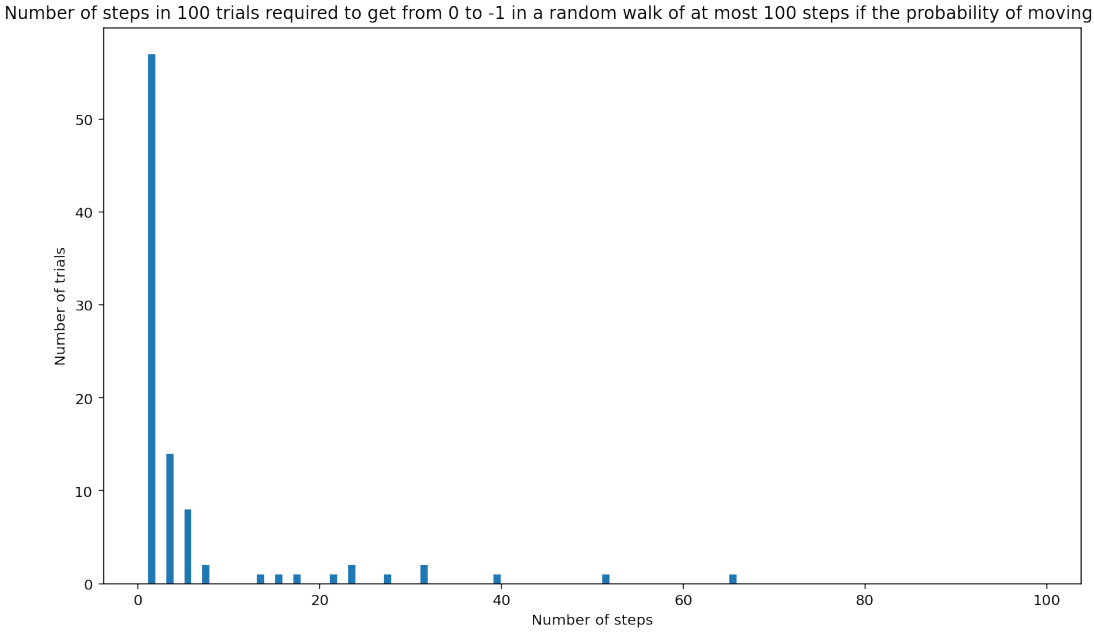

```
<Figure size 864x504 with 0 Axes>
```

This histogram shows how often each number 1-30 of heads was flipped. The most common number of heads appears to be around 5 or 6. It is very rare for this biased coin to land on heads for more than about 16 of 30 flips, so the heights of the bars are not visible for the number of heads between about 16 and 30. The graph has the shape of a binomial distribution with a shape very similar to the graph of the mathematical distribution given by the formula.

5.9.5 Random walks

Exercise 1: A person walks on a street with an equal probability of movement to the left or right at each pace. Write a Python program to approximate the expected number of paces the person should take in order to move one pace to the left of the starting point. Suggestion: Use 0 as the starting point. State your conclusions in a complete sentence.

```
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,␣
↪1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,␣
↪1, 1, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 5, 5, 5, 5, 5, 5, 5, 7, 7,␣
↪13, 15, 17, 21, 23, 23, 27, 31, 31, 39, 51, 65]

In 7 of 100 trials (7.000000000000001%), the position -1 was not reached after 100␣
↪steps.
```




```
<Figure size 864x504 with 0 Axes>
```

About half the time, it takes a person one step move one pace to the left of the starting point. This is because there is a probability of 0.5 of moving left initially (and at each step). Note that it always takes an odd number of steps to get to -1. Approximately 15% of the time, it takes three steps to get to -1. About half again as often it takes five steps. Occasionally it takes more, with path lengths 7, 9, 11, …, 99 if 100 is the maximum number of steps allowed. A small percent of the time, position -1 is not reached within 100 steps. It is not very meaningful to say what the expected number of steps is in general because the number of steps can be quite large occasionally, but we can say that about half the time we arrive at -1 in one step.

Exercise 2: Consider an unbiased random walk on a square lattice. Write a Python program to find approximate answers to the following questions:

(a) What is the expected number of a paces the person should take in order to visit the location one pace to the left of the starting point (0, 0)? Explain.

(b) How far away from the starting point (0,0) does a person end up after n steps of an unbiased random walk on a square lattice? Explain.

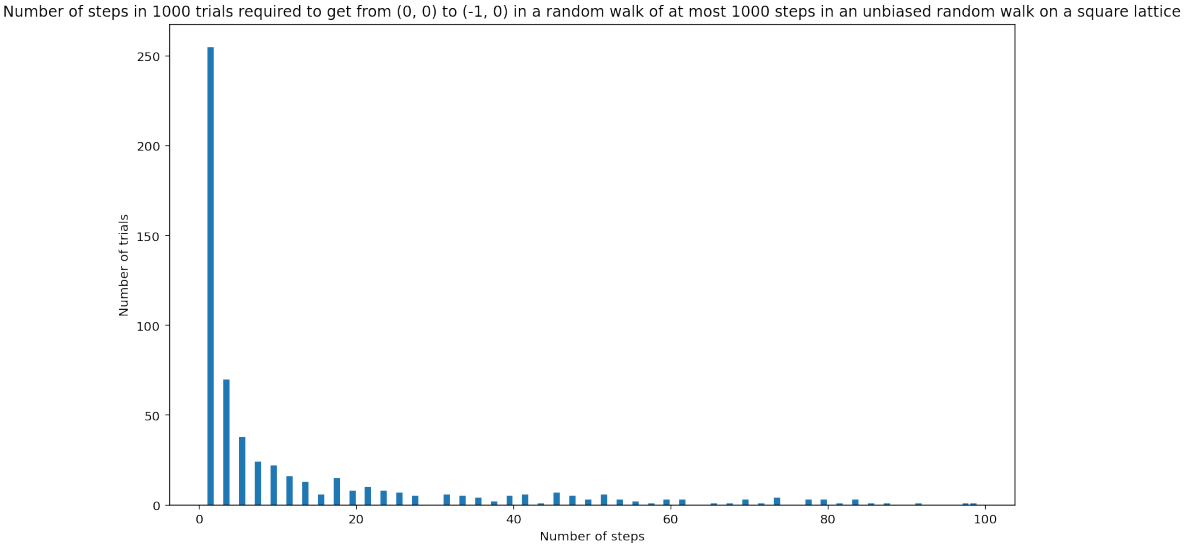
(c) If you run the simulation program for Part (b) many times, is there any statistical distribution you can expect? Explain.

```
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,␣
↪1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,␣
↪1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,␣
↪1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,␣
↪1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,␣
↪1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,␣
↪1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,␣
↪1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,␣
↪1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,␣
↪3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,␣
↪3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,␣
↪3, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,␣
↪5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,␣
↪7, 7, 7, 7, 7, 7, 7, 7, 7, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9,␣
```

(continues on next page)

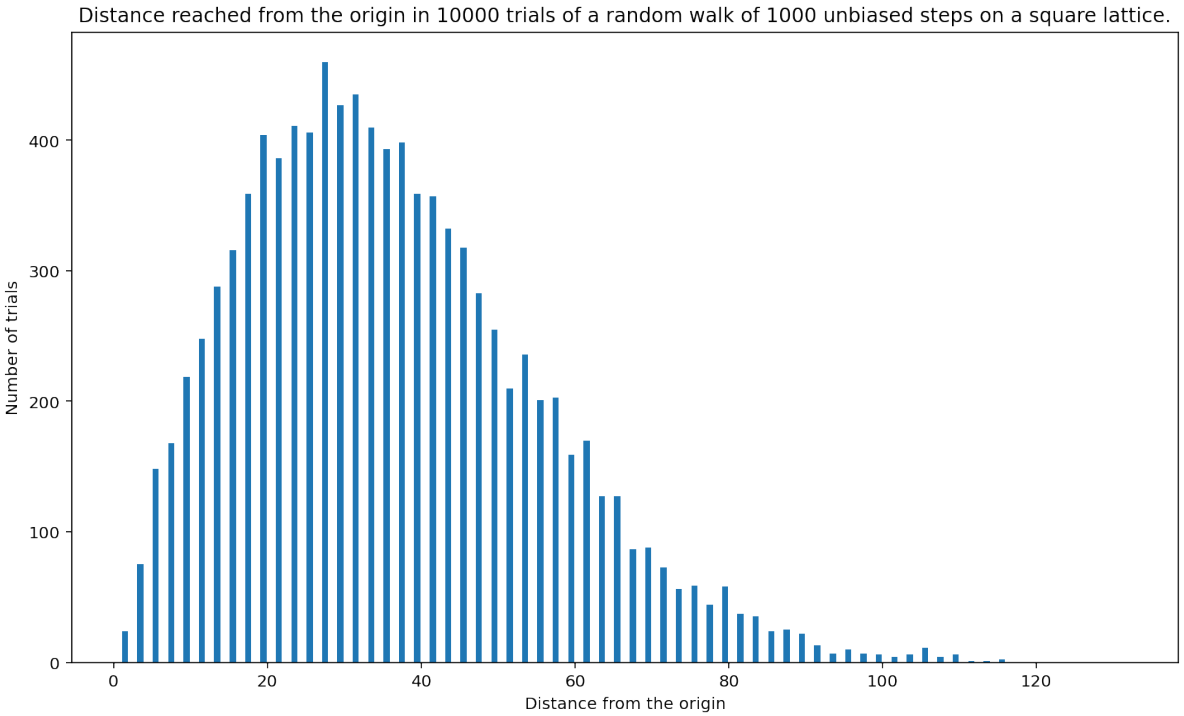
(continued from previous page)

```
 ↪9, 9, 9, 9, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 13,␣
 ↪13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 15, 15, 15, 15, 15, 15, 17, 17,␣
 ↪17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 19, 19, 19, 19, 19, 19, 19,␣
 ↪19, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 23, 23, 23, 23, 23, 23, 23, 23, 25,␣
 ↪25, 25, 25, 25, 25, 25, 27, 27, 27, 27, 27, 31, 31, 31, 31, 31, 31, 33, 33, 33,␣
 ↪33, 33, 35, 35, 35, 35, 37, 37, 39, 39, 39, 39, 39, 41, 41, 41, 41, 41, 41, 43,␣
 ↪45, 45, 45, 45, 45, 45, 45, 47, 47, 47, 47, 47, 49, 49, 49, 51, 51, 51, 51, 51,␣
 ↪51, 53, 53, 53, 55, 55, 57, 59, 59, 59, 61, 61, 61, 65, 67, 69, 69, 69, 71, 73,␣
 ↪73, 73, 73, 77, 77, 77, 79, 79, 79, 81, 83, 83, 83, 85, 87, 91, 97, 99]

In 416 of 1000 trials (41.6%), the position (-1, 0) was not reached after 1000␣
 ↪steps.

The person arrived at (-1, 0) in three steps 70 times.
```




```
<Figure size 864x504 with 0 Axes>
```

(a) About 25% of the time, it takes a person one step to move one pace to the left of the starting point. This is because there is a probability of 0.25 of moving left initially (and at each step). Note that it always takes an odd number of steps to get to (-1, 0). Approximately 7.5% of the time, it takes three steps to get to (-1, 0). About half again as often it takes five steps. Occasionally it takes more, with path lengths 7, 9, 11, …, 999, if 1000 is the maximum number of steps allowed. About 40% of the time, position (-1, 0) is not reached within 1000 steps. It is not very meaningful to say what the expected number of steps is in general because the number of steps can vary a lot, can be very large, and most commonly takes the person so far from (-1, 0) that our simulation gives up on having them return. However, we can say that about 25% of the time we arrive at (-1, 0) in one step.




```
<Figure size 864x504 with 0 Axes>
```

(b) If a person ends up at position (x, y), we can consider them to be a distance $|x| + |y|$ from the origin. In 100 steps, a person most commonly ends up about 10 "city blocks" from the origin. In 1000 steps, a person most commonly ends up about 30 units from the origin. (It can be shown mathematically that the distance the person ends up from the origin is proportional to $\sqrt{n}$ for long walks of n steps.)

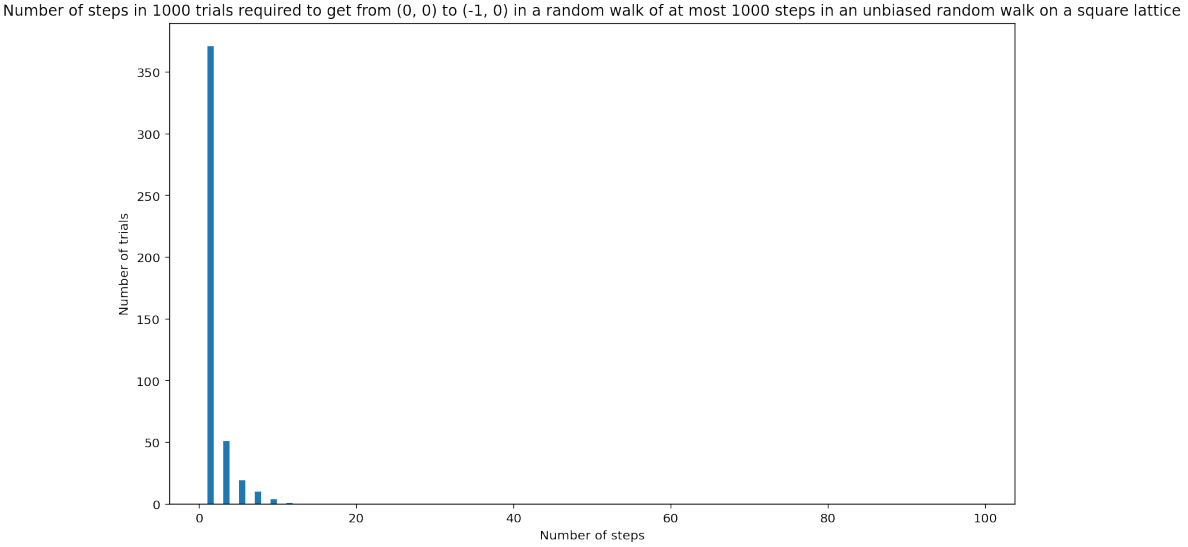
(c) The distribution of distances looks like a binomial distribution as shown.

Exercise 3: A person walks one pace left with a probability of 0.35, one pace right with a probability of 0.2, one pace forward with a probability of 0.4 and one pace backward with a probability of 0.05. Write a Python program to approximate the expected number of paces the person should take in order to visit the location one pace to the left of the starting point. State your conclusions in a complete sentence.

```
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,␣
↪
```

(continued from previous page)

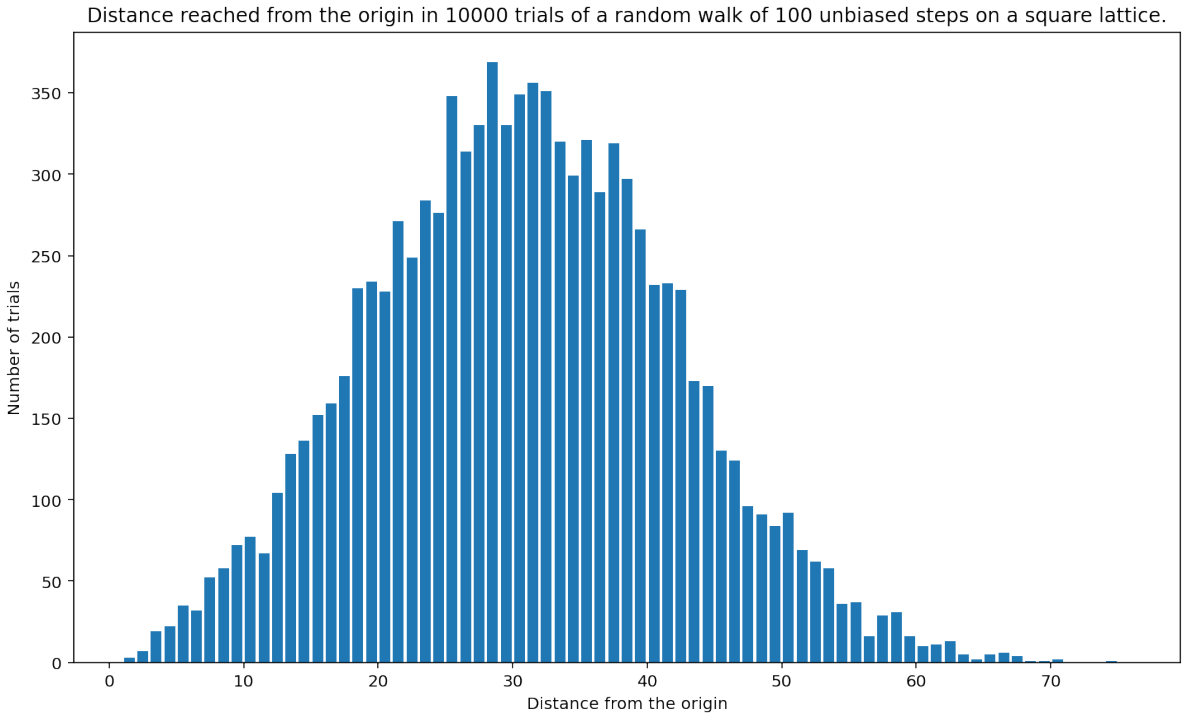
```
In 544 of 1000 trials (54.400000000000006%), the position (-1, 0) was not reached␣
↪after 1000 steps.
```




```
<Figure size 864x504 with 0 Axes>
```

About 35% of the time, it takes a person one step to move one pace to the left of the starting point. This is because there is a probability of 0.35 of moving left initially (and at each step). Note that it always takes an odd number of steps to get to (-1, 0). Approximately 5% of the time, it takes three steps to get to (-1, 0). About half again as often it takes five steps. Occasionally it takes more, with path lengths slightly longer like 7, 9, or 11. About 50% of the time, position (-1, 0) is not reached within 1000 steps. It looks like if the person doesn't reach (-1, 0) within a relatively small number of steps like 11, then they do not reach the position at all withing 1000 steps. It is not very meaningful to say what the expected number of steps is in general because the arrival time is typically either in one step or (for all practical purposes) never. We can certainly say that about 35% of the time we arrive at (-1, 0) in one step.

Exercise 4: A person walks one pace left, right, forward or back with equal probability. When the person walks left, right, or back, she walks one pace. When she walks forward, she walks two paces. Write a Python program to approximate how far forward the person has walked in 1000 paces. State your conclusions in a complete sentence.




```
<Figure size 864x504 with 0 Axes>
```

If a person ends up at position (x, y), we can consider them to be x units forward of the starting position (0, 0). In 100 steps, a person most commonly ends up about 27 units forward. In 1000 steps, a person most commonly ends up about 270 units forward. A histogram of distances forward has its peak skewed left as shown.

Exercise 5: A **Markov chain** describes a sequence of possible events in which the probability of each event depends on the state in the previous event. For example, suppose that we modify an unbiased random walk on a square lattice such that if the previous step was to the right, then the probability of making the current move up, down, left, or right is 0.2, 0.2, 0.4, or 0.2, respectively. As such,

(a) explain the implications of a move to the right in practical terms in a complete sentence.

(b) run some simulations and address the question: How does the use of the Markov chain change the unbiased random walk on a square lattice?

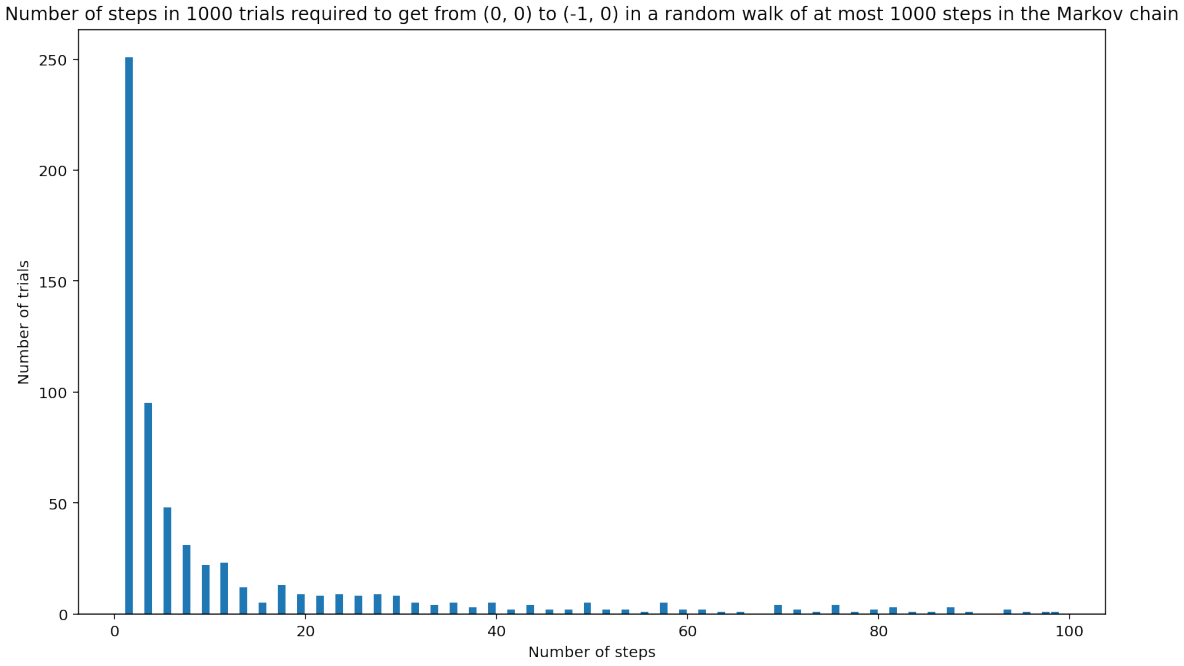
(a) In this problem, a move to the right makes the walker more likely to go left directly afterward.

```
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,␣
↪1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,␣
↪1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,␣
↪1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,␣
↪1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,␣
↪1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,␣
↪1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,␣
↪1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,␣
↪1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,␣
↪3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,␣
↪3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,␣
↪3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 5, 5, 5, 5, 5,␣
↪5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,␣
↪5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,␣
```

(continues on next page)

(continued from previous page)

```
↪7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 9, 9, 9, 9, 9, 9, 9,␣
↪9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 11, 11, 11, 11, 11, 11, 11, 11, 11,␣
↪11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 13, 13, 13, 13, 13, 13,␣
↪13, 13, 13, 13, 13, 13, 15, 15, 15, 15, 15, 17, 17, 17, 17, 17, 17, 17, 17, 17,␣
↪17, 17, 17, 17, 19, 19, 19, 19, 19, 19, 19, 19, 19, 21, 21, 21, 21, 21, 21, 21,␣
↪21, 23, 23, 23, 23, 23, 23, 23, 23, 23, 25, 25, 25, 25, 25, 25, 25, 25, 27, 27,␣
↪27, 27, 27, 27, 27, 27, 27, 29, 29, 29, 29, 29, 29, 29, 29, 31, 31, 31, 31, 31,␣
↪33, 33, 33, 33, 35, 35, 35, 35, 35, 37, 37, 37, 39, 39, 39, 39, 39, 41, 41, 43,␣
↪43, 43, 43, 45, 45, 47, 47, 49, 49, 49, 49, 49, 51, 51, 53, 53, 55, 57, 57, 57,␣
↪57, 57, 59, 59, 61, 61, 63, 65, 69, 69, 69, 69, 71, 71, 73, 75, 75, 75, 75, 77,␣
↪79, 79, 81, 81, 81, 83, 85, 87, 87, 87, 89, 93, 93, 95, 97, 99]

In 368 of 1000 trials (36.8%), the position (-1, 0) was not reached after 1000␣
↪steps.

The person arrived at (-1, 0) in three steps 95 times.
```




```
<Figure size 864x504 with 0 Axes>
```

(b) It is still the case that approximately 25% of the time the person needs one step to move one pace to the left of the starting point. This is because there is a probability of 0.25 of stepping to the left on the initial step. However, the Markov chain makes arrival at (-1, 0) in 3 steps more common than before. (It accounts for more like 9% of the trials, rather than 7.5%.) Similarly, taking 5, 7, 9, 11, …, or 999 steps to reach (-1, 0) is also more common. This is because every time the person steps to the right, there is a higher probability of going left (0.4) next, as compared with 0.25 in the non-Markov chain problem. As a result, it is less common (maybe 38% of the time on average rather than 40%) that the position (-1, 0) is not reached within 1000 steps.

Exercise 6: Two people get lost in a shopping mall. They both want to find each other as soon as possible. However, each has no clue where the other one is. There are two strategies available:

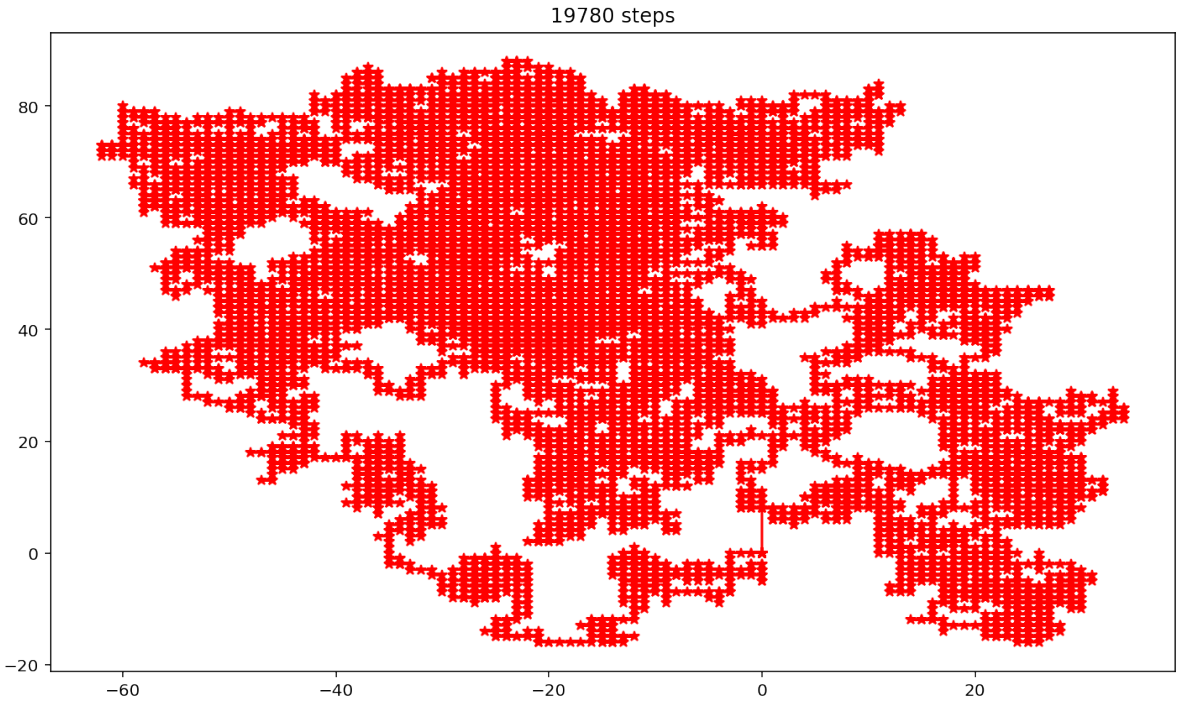
(1) One person stays in the same place. The other moves around.

(2) Both people move.

Which strategy is better for them to meet?

Do simulations and explain how they support your answer.

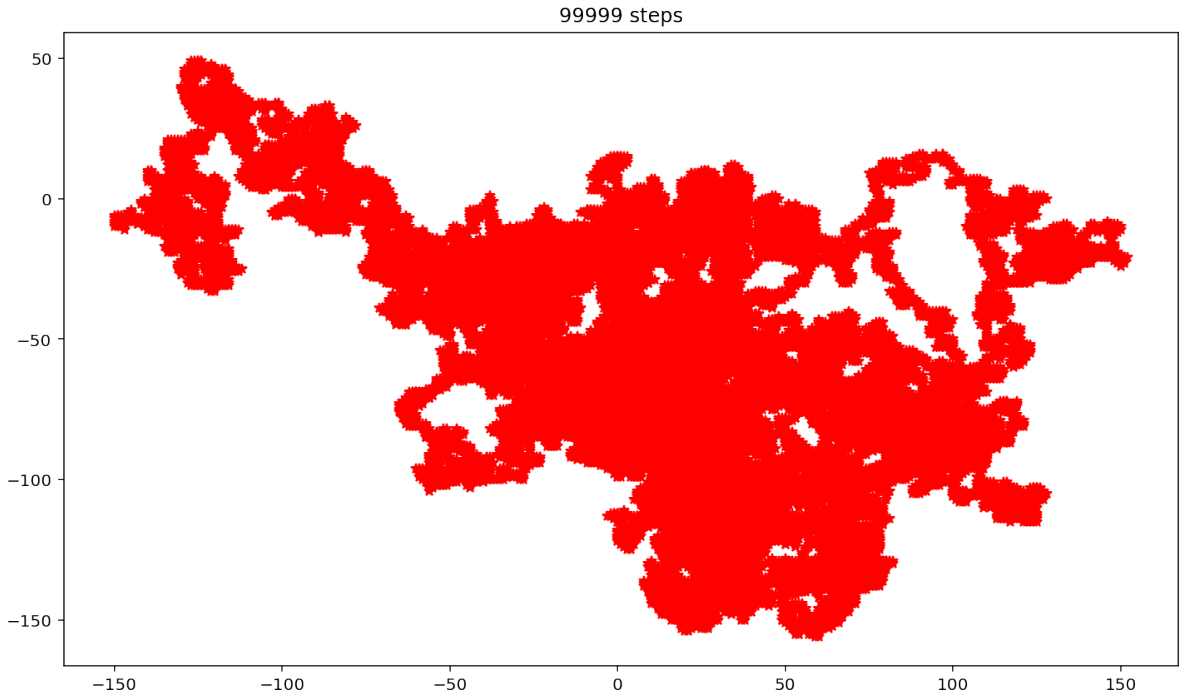
```
Target found in 19780 steps.
```




```
Final position
```

```
(0.0, 10.0)
```

```
Target not found in 100000 steps.
```




```
Final position
```

```
(-120.0, -3.0)
```

Simulations on two persons that start ten paces apart show that typically the better strategy is for one person to stay in the same place and for the other to move around. Simulations of this strategy usually end with the two people finding each other. Simulations in which both people engage in random walks typically end with them failing to meet within 100,000 steps.

5.9.6 Agent-based models

Exercise 1: Simulate the spread of information under the given assumptions. How many days does it seem to take in general for the information to spread to half of the population? Will all individuals be informed after a certain number of days?

```
def rumor_spread(total_population):
    # Set a maximum number of iterations.
    max_it = 1000
    # Inform Agent 0.
    informed_status = [1]
    # No one else has been informed.
    for i in range(total_population-1):
        informed_status.append(0)
    # Generate a random number from 0 to 5 for each agent and store in a list as the␣
↪number of transmissions.
    transmission_numbers = []
    for i in range(total_population):
        transmission_numbers.append(random.randint(0, 5))
    # Simulation of the process
    day = 0
```

(continues on next page)

(continued from previous page)

```
    spread_to_half = False
    day_spread_to_half = 0
    while informed_status.count(1) != total_population and (day < max_it):
        # Record the first day that at least half of the population has the␣
↪information.
        if not(spread_to_half) and informed_status.count(1) >= int(total_population/
↪2):
            day_spread_to_half = day
            spread_to_half = True
        new_status = informed_status[:]
        for agent_index in range(total_population):
            # print(informed_status)
            if informed_status[agent_index] == 1:  # If agent is informed,
                # transmit to (inform) the appropriate number of randomly selected␣
↪agents (excluding self).
                for informing in range(transmission_numbers[agent_index]):
                    recipient_index = random.randrange(total_population)
                    while recipient_index == agent_index:
                        recipient_index = random.randrange(total_population) # Ensure␣
↪agent does not transmit to self.
                    new_status[recipient_index] = 1
        # Copy the lists to ensure that informed_status and new_status point to two␣
↪different lists in memory.
        informed_status = new_status[:]
        day += 1
    if informed_status.count(1) == total_population:
        print(f"All the individuals are informed by Day {day}.")
    else:
        print(f"Not all the individuals were informed within the maximum number of␣
↪iterations ({max_it}).")
    if day_spread_to_half > 0:
        print(f"(Half of the individuals are informed by Day {day_spread_to_half}.)")

rumor_spread(1000)
```

Typically it seems to take about 6 days in general for the information to spread to half of the population of 1000 and about 9 days for all individuals to be informed. However, sometimes not all the individuals are informed within a maximum number of iterations, such as 1000. Upon closer inspection (by doing print(informed_status.count(1) upon exiting the while loop), this situation seems to arise when the initial agent is assigned a transmission number of 0.

```
All the individuals are informed by Day 8.
(Half of the individuals are informed by Day 5.)
```

Typically it seems to take about 6 days in general for the information to spread to half of the population of 1000 and about 8 days for all individuals to be informed. However, sometimes not all the individuals are informed within a maximum number of iterations, such as 1000. This situation arises when the transmission numbers are too low to produce the typical pattern of information spread.

Exercise 2: This agent-based model could also be applied to the spread of disease. If "knowing the information" means "having the disease" and "not having the information" means "not having the disease", what are the interpretations of the assumptions, initial condition, and parameters that are given? Explain how you may find the model compelling in the context of disease, or what changes to the model you recommend to better reflect the spread of disease.

If a disease is shared, it might spread from one agent to another similarly to how information spreads. We can assume, similarly to before that:

1. The population is a closed community. The number of individuals does not change. With a disease raging, this

assumption may not be sound.

2. Agents who do not have the disease do not pass the disease to others. This assumption is not good for all diseases. During COVID, we saw that asymptomatic individuals could spread the disease. So people who appeared "not sick" were still contagious.
3. An agent does not pass the disease to itself.
4. Once an agent catches the disease, it will not be uninfected. This assumption may not be good, as there should be a transition from "sick" to "not sick".
5. A transmission rate defines how many people catch the disease from a given individual in one day.
6. Each agent has one of the allowed transmission rates with equal probability. It is unclear whether different people have different degrees of contagion.
7. When an agent spreads the disease to another individual, that individual is selected from all the other members of the population with equal probability. This might be no less contrived than the analogous assumption in the information-spreading model. However, in reality, both cases involve social networks in which some people come into contact with a given individual more than others.

Like for the information spreading, we could impose the initial condition that on Day 0, 1 individual is sick. (This person is called the "index case" in the field.)

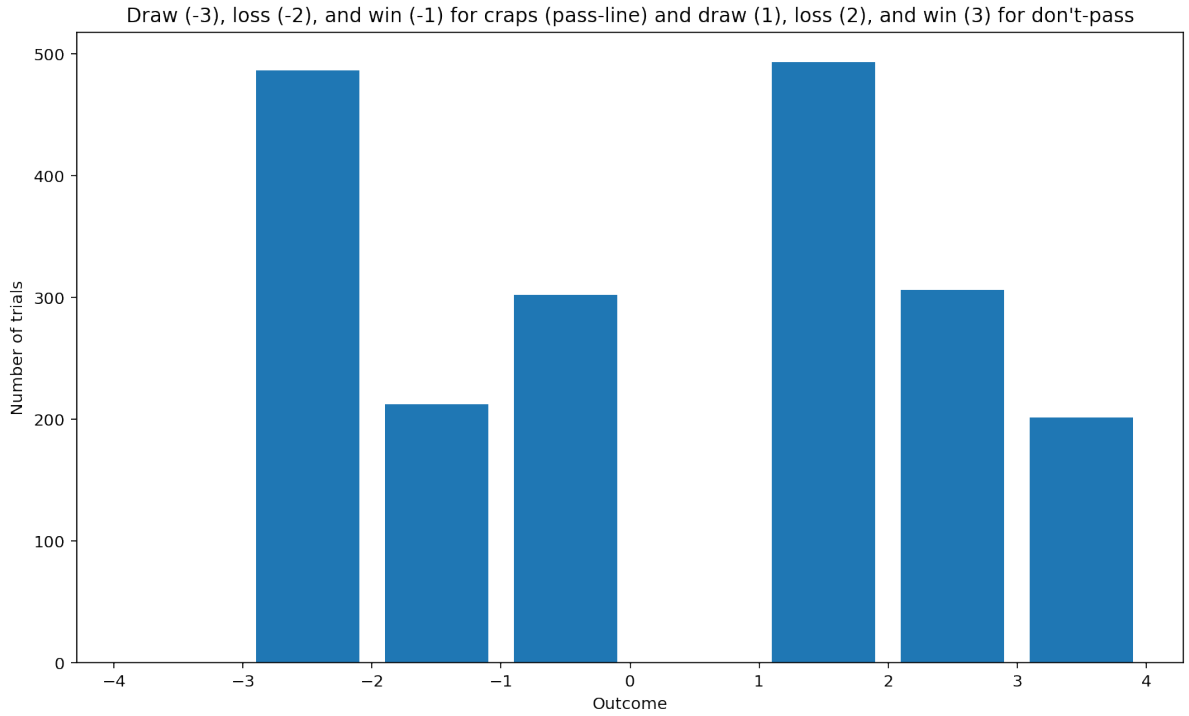
For parameters for the agent-based model of disease transmission, we could again say

1. The value of the population parameter is 1000. (There are 1000 agents.)
2. The allowed transmission rates are 0, 1, 2, 3, 4, and 5.

The way the disease could spread is as follows: We have Persons 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,…, 1000. These people might have transmission rates 3, 3, 2, 0, 5, 1, 1, 2, 3, 4, 2, 1, …, 4 respectively. Initially, 1 person is sick. If that person is Person 1, and Person 1 has a transmission rate of 3, then on the first day of illness, Person 1 might spread the disease to Persons 5, 8, and 9 with transmission rates 5, 2, and 3, respectively. In Day 2 of the pandemic, Person 1 spreads the information to 3 people, Person 5 to 5 people, Person 8 to 2 people and Person 9 to 3 people. Note these may be some of the same people or people who already are sick in our model. The process continues day-by-day. One of the complicating factors with COVID was that people could spread the disease a couple of days before they showed symptoms themselves. Our model does not account for this.

5.9.7 Mathematical Games

Exercise 1: For the game of craps, does the choice of "Pass Line" vs. "Don't Pass" affect the likelihood of winning, under the rules given? If so, what kind of difference and how much difference? Use quantitative information to support your answer. State whether you have provided mathematical proof or computational evidence.

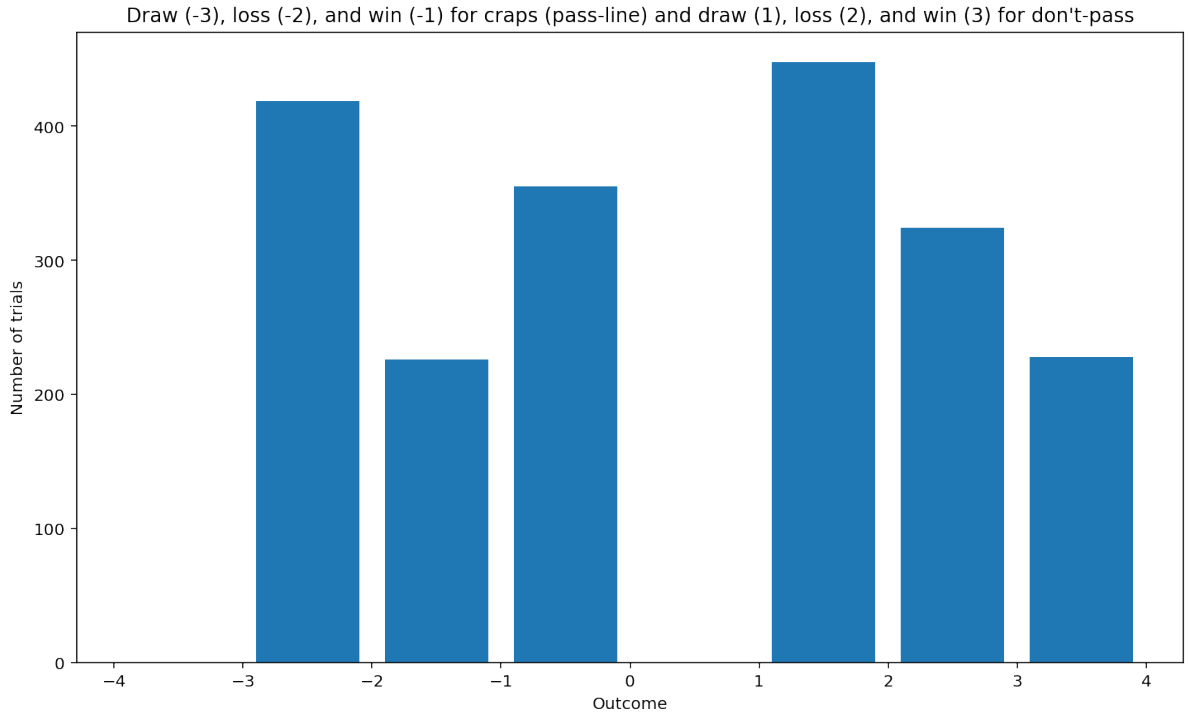



```
<Figure size 864x504 with 0 Axes>
```

The results give computational evidence that the "Pass Line" rules are favorable. In both "Pass" and "Don't Pass" the most common outcome is a draw. But for losses then wins, respectively, we see the third bar higher than the second in the first set of three bars, meaning that there are more wins than loses. The "Don't Pass" rules make a win less likely than a loss: the third bar is lower than the second in the second set of three bars. It appears that the number of wins for "Pass" are comparable to the number of losses for "Don't Pass", and the number of losses for "Pass" are comparable to the number of wins for "Don't Pass".

Exercise 2: If one of the two dice is biased in the game of craps, how does that affect the analysis and conclusions from Exercise 1? Illustrate your answer by doing some simulations and discussing the results.

```
histogram_craps(1000, False)
```




```
<Figure size 864x504 with 0 Axes>
```

The qualitative comparison between the outcomes of the two versions of the game is quite robust to various biases in the dice for the biases that I tried. There are quantitative changes in the outcomes. For example, a win becomes more likely for don't pass than before if one of the dice rolls a 5 with probability 1/4, a six with probability 1/4, and a 1, 2, 3, or 4 with probability 1/8 each. However, wins are still more common under the pass-line rules than don't-pass even in this case. Draws are less frequent than with unbiased dice, but they are still the most frequent outcome.

Exercise 3: Simulate the optimal stopping problem for ten rooms using the algorithm discussed in this section. Execute many trials.

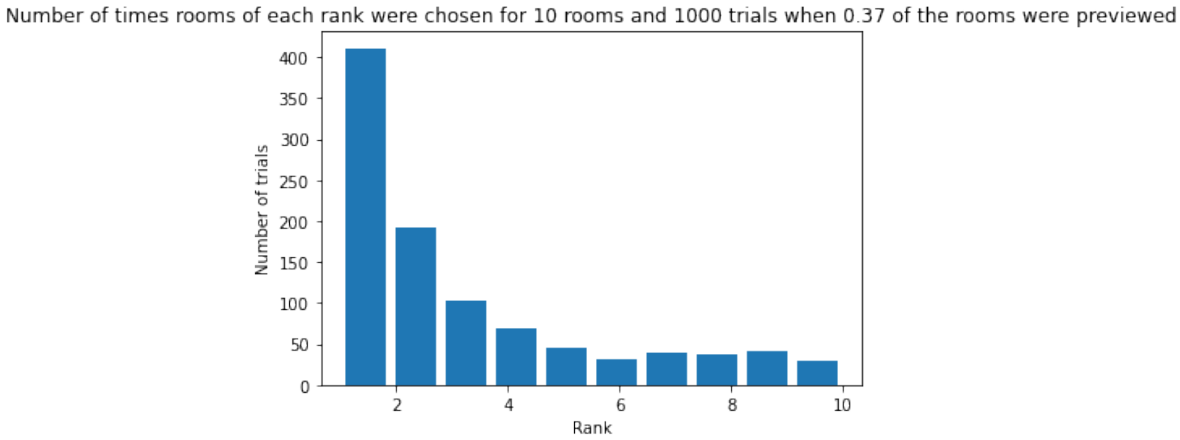
(a) Explain whether you typically get the best room or not when you preview 37% (three or four) of the rooms.

(b) If you don't get the best room, how good of a room do you typically get?

(c) Show a histogram of the ranks of the rooms you choose. Do the shape, center, and spread of the ranks suggest that the strategy (algorithm) is a good one for winning the best room? Explain.

(d) Does the strategy of previewing three or four of the ten rooms tend to lead to a better outcome than if you preview two or five? Support your answer.

```
Number of rooms: 10
Number of rooms previewed: 3
Number of trials with 10 rooms per trial: 1000
Percent of times best room was picked: 41
Percent of times second best room was picked: 19
Percent of times third best room was picked: 10
Percent of times one of the top three rooms was picked: 71
Percent of times worst room was picked: 3
```

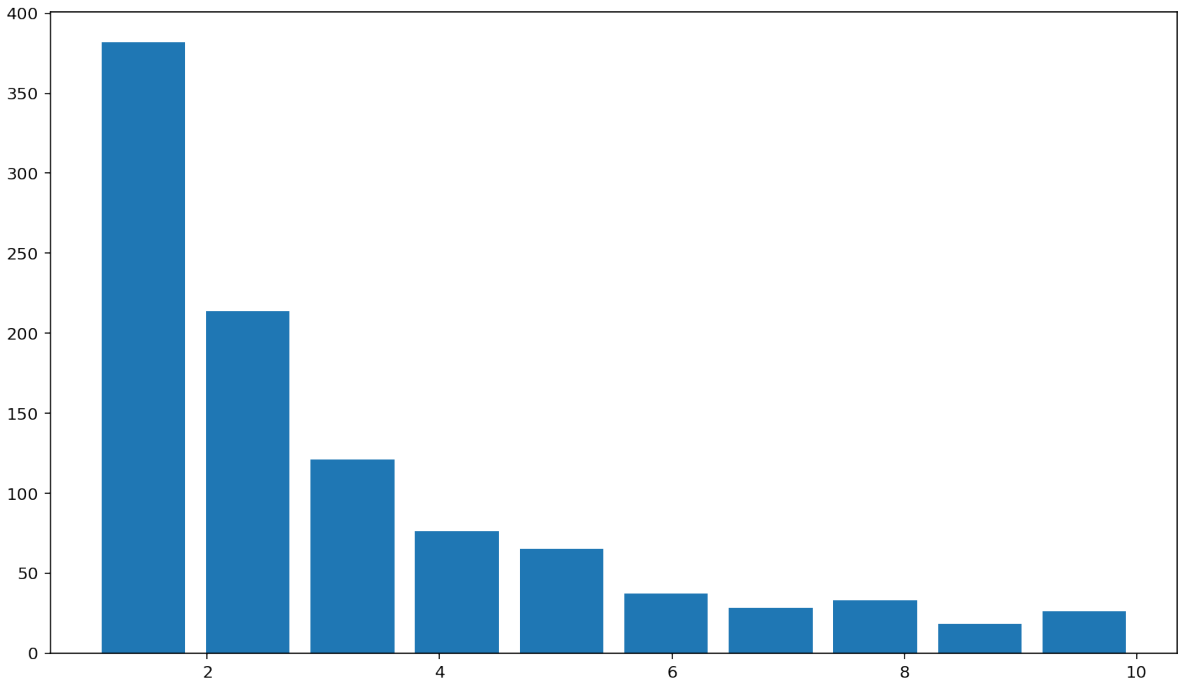


(a) When you preview three of the rooms, you get the best room about 41% of the time, which is a lot better than a random choice of $1/n = 1/10$.

(b) You typically get a room in the top three about 70% of the time—about 20% of the time you get the second best and about 10% of the time you get third best.

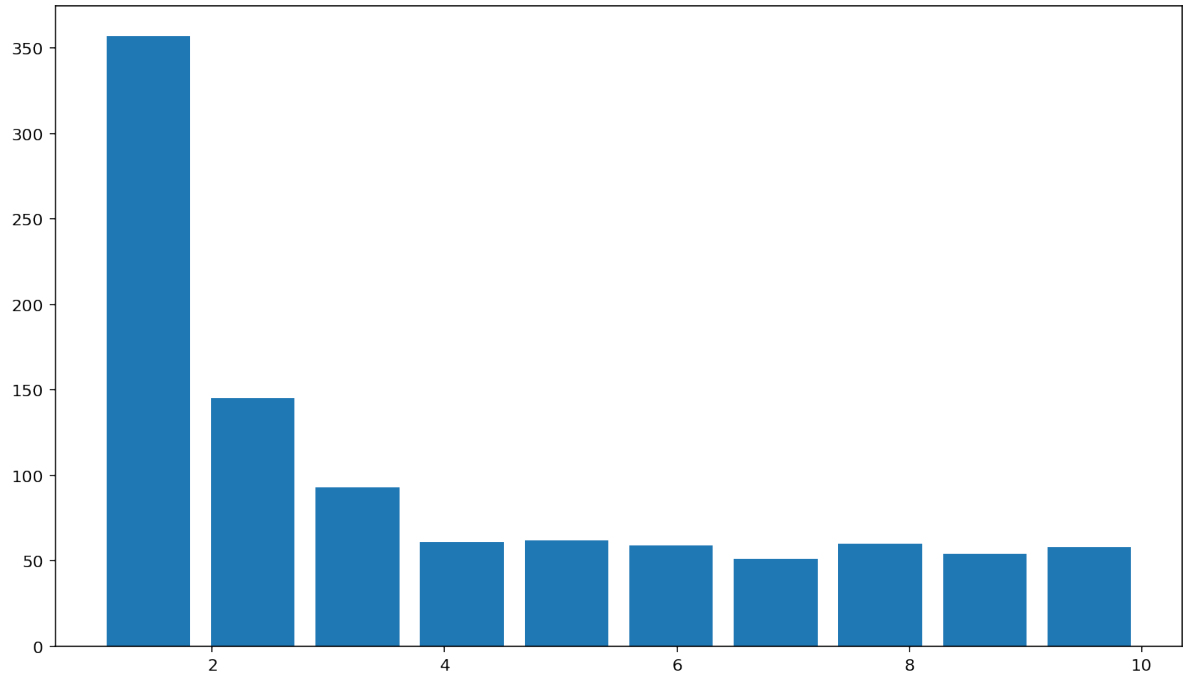
(c) The histogram suggests that the strategy (algorithm) is a good one for winning the best room because it shows that Rank 1 is the dominant rank selected, and lower and lower ranks are generally chosen less and less often.

```
Number of rooms: 10
Number of trials with 10 rooms per trial: 1000
Percent of times best room was picked: 38
Percent of times second best room was picked: 21
Percent of times third best room was picked: 12
Percent of times one of the top three rooms was picked: 72
Percent of times worst room was picked: 3
```



```
trials_of_room_visits(1000, 10, 0.5)
```

```
Number of rooms: 10
Number of trials with 10 rooms per trial: 1000
Percent of times best room was picked: 36
Percent of times second best room was picked: 14
Percent of times third best room was picked: 9
Percent of times one of the top three rooms was picked: 60
Percent of times worst room was picked: 6
```



(d) The strategy of previewing three of the ten rooms tends to lead to a better outcome than if you preview a fraction 0.2 of the rooms (two rooms) or 0.4 of the rooms (four rooms). With a preview of two rooms, the percentage of times that you get the best room drops slightly compared to a preview of three rooms. The percentage of times you get a room in the top three seems to hold fairly steady. With a preview of four rooms, the percentage of times that you get the best room drops a little compared to a preview of three rooms. The percentage of times you get a room in the top three drops quite noticeably. With a preview of four rooms, the histogram shows a slightly lower peak at Rank 1 and a thicker tail for the distribution, meaning lower-rank choices are more common.

(e) These results are generally consistent with the mathematical finding that the optimal fraction of the total rooms that you should preview is $1/e$ because $1/e$ is between 30 and 40% of the rooms. Technically, this mathematical conclusion only applies as n gets very large, so it is striking that even when n is only 10, we should still preview around 37% of the rooms. If the probability that you will select the best room this way tends to $1/e$ for large n, then we would predict obtaining the best room about 37% of the time. Our simulations show slightly better results than that when we preview three or four out of the ten rooms. It's reasonable to imagine that getting the best room might become harder as n gets large, so probably if we simulate with a larger number of rooms and preview about 37% of them, we might get the best room only about 37% of the time, rather than 40 or 41%.

Exercise 4: With a fair coin, is there a winning strategy to maximize your points in the coin-flipping game described in this section? If so, what is the winning strategy? Illustrate your conclusions using simulation.

Exercise 5: If you know the coin is biased, with 60% chance of heads and 40% chance of tails, how should you change your strategy? Illustrate your answer using simulation.

```
Game ended with 0 <= points < 1 approximately 50% of the time within a maximum␣
 ↪number of turns = 30.
Game ended with 1 <= points <= 100 initial tokens approximately 42% of the time.
So, the player won additional tokens approximately 8% of the time.
The average loss between 1 and 100 across all trials is approximately 93 points.
The average winnings (meaning average number of final_points beyond the initial␣
 ↪100) across all trials is approximately 910.
Maximum score was approximately 39455 in 10000 trials.
```

With a fair coin, the formula in the chapter says the optimal F is 0, meaning that there is no winning strategy to maximize your points in the coin-flipping game described in this section. The simulation shows that about 50% of the time you lose all your tokens (100 tokens) within the maximum number of turns in the game (30). You end the game with more points than you started with only about 8% of the time. Note that you consistently lose 93 of your 100 tokens when you lose. The average winnings when you do win tend to be about 1000 tokens. The number of points you have at the end can occasionally be very large. A high score is possible but improbable, within this strategy that I do not consider a winning strategy.

Exercise 5: If you know the coin is biased, with probability 0.6 of success, the formula in the chapter says you should use $F = 0.2$.

```
flipping_game(0.6, 100, 0.2, 30, 10_000)
```

```
Game ended with 0 <= points < 1 approximately 0% of the time within a maximum␣
 ↪number of turns = 30.
Game ended with 1 <= points <= 100 initial tokens approximately 34% of the time.
So, the player won additional tokens approximately 66% of the time.
The average loss between 1 and 100 across all trials is approximately 36 points.
The average winnings (meaning average number of final_points beyond the initial␣
 ↪100) across all trials is approximately 382.
Maximum score was approximately 8440 in 10000 trials.
```

This strategy has a very different outcome. When we play the game this way, we never seem to lose all the tokens (instead of losing all of them half the time). Your winnings are less on average (around 380) this way than with the fair coin and $F = 1/2$. However, you have more points at the end that at the beginning far more often— around 66% of the time rather than 8%. When you do lose, the losses are less on average (35 points, rather than 93 points).

5.10 JNB Lab: Homicides in Chicago

This lab is an outcome of a collaboration in 2019-2021 between Paul Isihara (Professor of Mathematics, Wheaton College (IL)) and a team at Flourishing Community Initiative of Sunshine Gospels Ministries which included Arnold Sojourner (FCI director), Joel Hamernick (SGM Director) and Evan Trowbridge (FCI Data Analyst) Dr. Paul Campbell (Editor, UMAP Journal) served as a statistical consultant.

Parts of this material was published previously in an article in the UMAP Journal titled Data Detectives: Homicides in Chicago. The UMAP Journal 42(1) (2021) 27-33. Copyright 2021 by COMAP, Inc. All rights reserved. Authosr: Paul Isihara. Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice. Abstracting with credit is permitted, but copyrights for components of this work owned by others than COMAP must be honored. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior permission from COMAP.

5.10.1 Historical Context

Chicago has suffered over 8,000 homicides in the past 20 years, or more than 1 homicide per day on average. A number of historical factors have contributed to the current level of homicides in Chicago.

After slavery was abolished, racism in the form of segregationist Jim Crow Laws and white supremacy led by the KKK contributed to the Great Migration of 6,000,000 black Americans who left their homes in the South and moved in large numbers to cities such as Chicago. Further racism occurred in the form of hatred in the industrial workplace, redlining which kept black Americans from home ownership, slum landlording which created sub-standard and overcrowded rental and school conditions, and gentrification which further displaced and forced black Americans into communities which experienced white flight, dis-investment, and loss of jobs. As hope of a good education, honest employment, and decent living conditions disappeared, some were led into gangs and drugs. Mass incarceration further exacerbated the family instability, economic hardship, and violence of predominantly black American areas of the city.

5.10.2 DATA VISUALIZATION

1) Read in Homicide data for Cook County which was obtained from the Cook County Medical Examiner Case Archive https://datacatalog.cookcountyil.gov/Public-Safety/Medical-Examiner-Case-Archive/cjeq-bs86 on Feb 6, 2021, and stored in an Excel file homicides.xlsx. Drop rows that have missing data.

```
     Date of Incident   Age Gender   Race            Primary Cause  \
0 2017-02-26 10:48:00  23.0   Male  Black  MULTIPLE GUNSHOT WOUNDS

  Residence City  Incident Zip Code
0        Chicago            60623.0
```

2) Extract the month and year of homicide incidents.

```
     Date of Incident   Age Gender   Race            Primary Cause  \
0 2017-02-26 10:48:00  23.0   Male  Black  MULTIPLE GUNSHOT WOUNDS

  Residence City  Incident Zip Code    new_date  year  month  day
0        Chicago            60623.0  2017-02-26  2017      2   26
```

3) Get Chicago data, select the columns [“Age”,“Gender”,“Race”,“Primary Cause”,“Incident Zip Code”,“year”,“month”,“day”] and use integer format

```
   Age Gender   Race            Primary Cause  Incident Zip Code  year  month  \
0   23   Male  Black  MULTIPLE GUNSHOT WOUNDS              60623  2017      2

   day
0   26
```

4) Get the homicide totals by year

```
2016    669
2020    620
2017    566
2018    492
2019    433
2015    429
Name: year, dtype: int64
```

5) Sort the data by year

6) Plot the Homicides by Month

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

5.10.3 Monte Carlo Analysis

The Monte Carlo method uses pseudo-random numbers for statistical analysis. We will use this approach to create a histogram of homicide counts for 10,000 hypothetical months of May based on probabilities of daily homicide counts using actual data for the years 2015-2019. For example, we first record that May 1, 2015 had 3 homicides, May 2, had 0 homicides,…, May 31, 2019 had 5 homicides, and then make a frequency distribution called "maycount" using the numbers 3,0, …, 5. We obtain in this way the empirical probability that there will be 0 homicides on a day in May is 0.219 since 21.9% of the days in May (for 2015-2019) had 0 homicides. We then use the empirical probabilities and pseudo-random numbers tto generate a random sample of 31 homicide counts for May, and these daily counts to get a random monthly total for May. We repeat this 10,000 times and make a histogram of the results.

1) Get the number of Chicago homicides in May 2020.

```
Chicago Homicides In May 2020:  77
```

2) Get the number of homicides which occurred on each day of May between 2015 and 2019.

```
0    5
1    0
2    1
3    0
4    2
dtype: int32
```

3) Get frequency distributions for the number of homicides in a day

```
0    0.200000
1    0.303226
2    0.219355
3    0.141935
4    0.083871
5    0.051613
dtype: float64
```

4) Do a Monte Carlo simulation for monthly homicide counts in May based on 31 random draws from the respective empirical distributions.

5) Make a histogram of the frequency distribution for May.

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
Text(0.5, 1.0, '10,000 Simulated May Homicide Counts based on 2015-2019 Data')
```

6) Find the empirical p-value of getting at least 77 homicides (the number in May 2020.)

```
Probability that a random draw of 31 days from May 2015-2020 distibution results␣
↪in 59 or more homicides: 0.0038
```

Exercises

1) a) Repeat the above analysis to obtain a histogram for July.

b) Find the empirical p-value of getting at least 108 homicides (the number in July 2020.)

2) a) Make a histogram of the difference July homicide count - May homicide count.

b) Find the probability that July will have at least n more homicides than May $(0 \leq n \leq 20)$. Make a plot of these probabilities.

c) Estimate the probability that there will be 20 more homicides in July than in May.

5.11 JNB Lab Solutions

Exercise 1a

1) Read in Homicide data and drop rows that have missing data.

```
     Date of Incident   Age Gender   Race            Primary Cause  \
0 2017-02-26 10:48:00  23.0   Male  Black  MULTIPLE GUNSHOT WOUNDS

  Residence City  Incident Zip Code
0        Chicago            60623.0
```

2) Extract the month and year of homicide incidents.

```
     Date of Incident   Age Gender   Race            Primary Cause  \
0 2017-02-26 10:48:00  23.0   Male  Black  MULTIPLE GUNSHOT WOUNDS

  Residence City  Incident Zip Code    new_date  year  month  day
0        Chicago            60623.0  2017-02-26  2017      2   26
```

3) Get Chicago data, select the columns ["Age","Gender","Race","Primary Cause","Incident Zip Code","year","month","day"] and use integer format

```
   Age Gender   Race            Primary Cause  Incident Zip Code  year  month  \
0   23   Male  Black  MULTIPLE GUNSHOT WOUNDS              60623  2017      2

   day
0   26
```

4) Get the number of Chicago homicides in July 2020.

```
Chicago Homicides In July 2020:  89
```

5) Get the number of homicides which occurred on each day of July between 2015 and 2019.

```
0    0
1    1
2    3
3    1
4    4
dtype: int32
```

6) Get frequency distributions for the number of homicides in a day

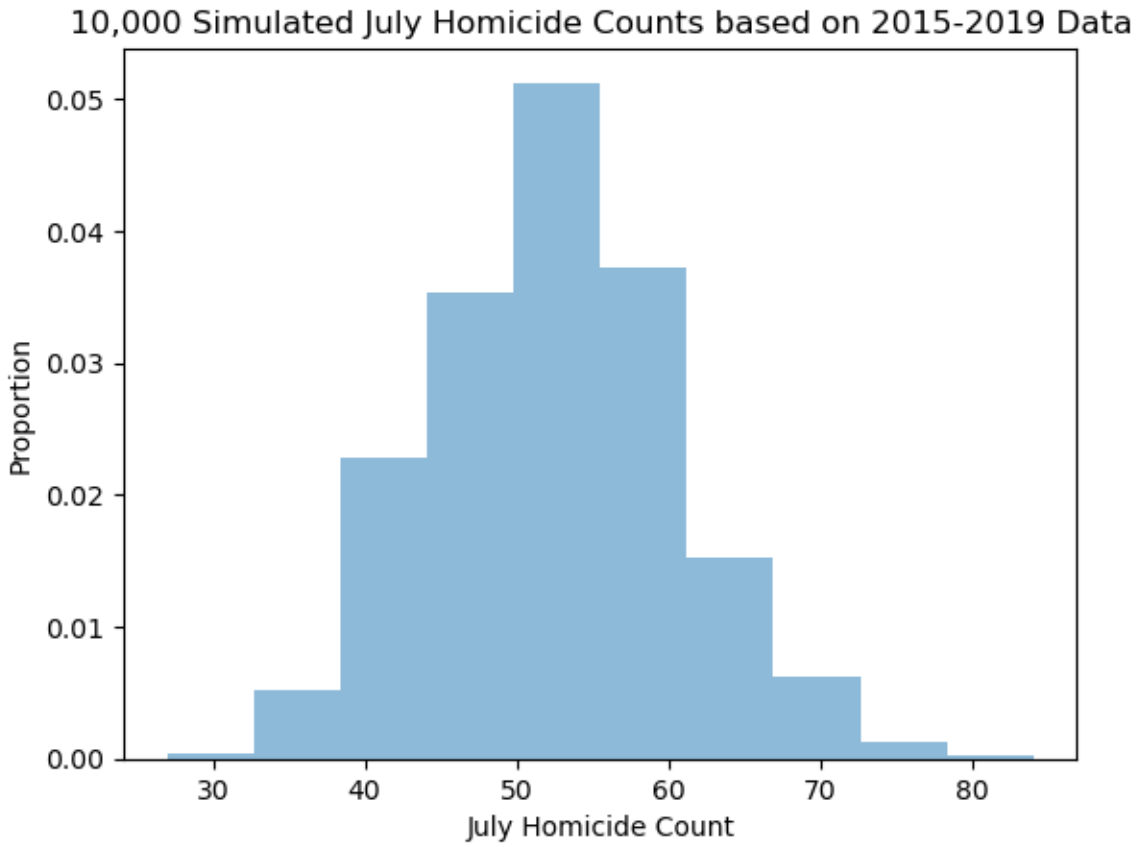
```
0    0.200000
1    0.309677
2    0.258065
3    0.141935
4    0.051613
5    0.012903
6    0.012903
7    0.012903
dtype: float64
```

```
0.2
```

7) Do a Monte Carlo simulation for monthly homicide counts in July based on 31 random draws from the respective empirical distributions.

8) Make a histogram for July.

```
Text(0.5, 1.0, '10,000 Simulated July Homicide Counts based on 2015-2019 Data')
```



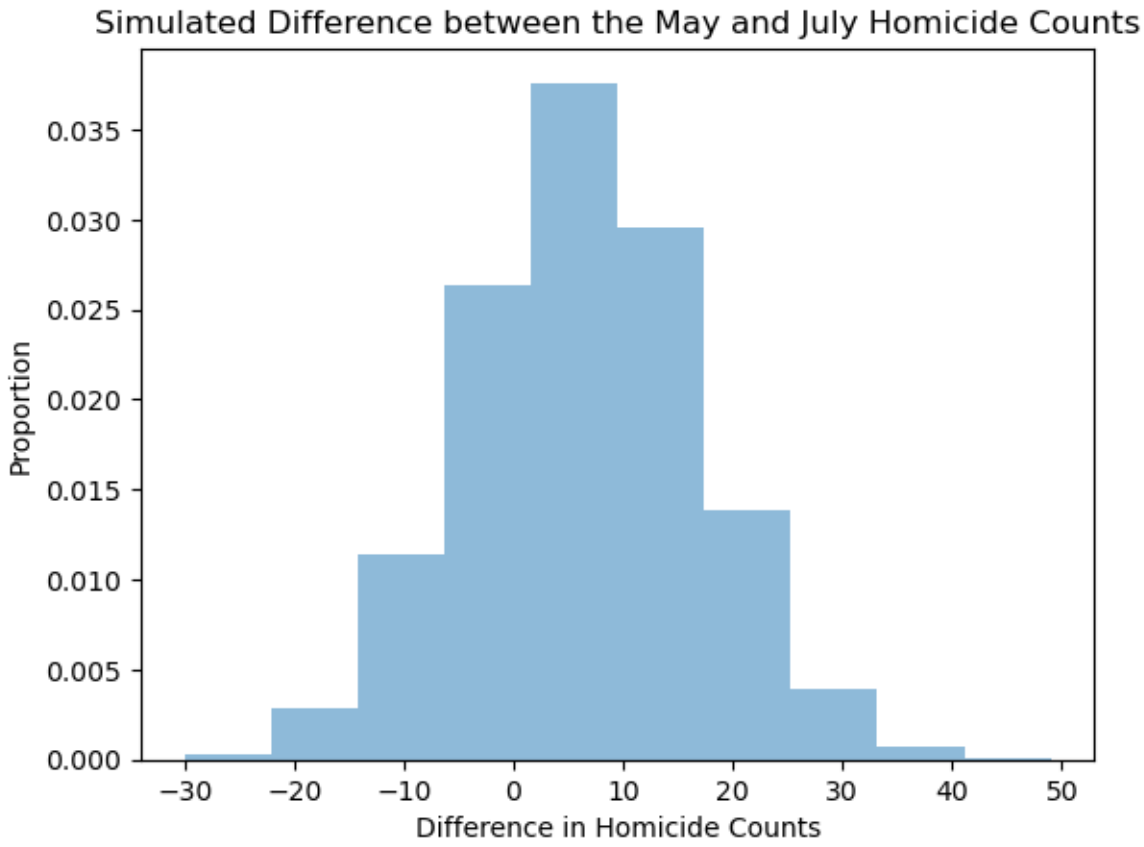
Exercise 1b

b) Find the empirical p-value of getting at least 108 homicides (the number in July 2020.)

```
Probaility that a random draw of 31 days from July 2015-2020 distibution results␣
↪in 85 or more homicides: 0.0
```

Exercise 2

a) Make a histogram of the difference July homicide count - May homicide count



Find the probability that July will have at least n more homicides than May $(0 \leq n \leq 20)$.

```
0     0.7409
1     0.7091
2     0.6766
3     0.6436
4     0.6077
5     0.5694
6     0.5298
7     0.4885
8     0.4529
9     0.4168
10    0.3796
11    0.3436
12    0.3082
13    0.2773
14    0.2459
15    0.2171
16    0.1889
17    0.1677
18    0.1461
19    0.1246
20    0.1045
dtype: float64
```

b) Make a plot of these probabilities

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

c) Based on 2015-2019 data, there is a 10% chance of 20 more homicides in July than May.

CHAPTER 6

STATISTICAL INFERENCE - Peter Jantsch, Claire Wagner

Note

Peter Jantsch and Claire Wagner will cover the basics of statistical inference.

6.1 Introduction

6.1.1 About This Chapter

In this chapter we introduce the basic tools of statistical inference: confidence intervals, hypothesis tests, and linear regression. Our goal is to show how each of these can be more fully understood using the computational tools of Python and Jupyter notebooks.

We have chosen to write the following sections as an interactive demonstration of the tools of statistical inference for students. Rather than a simple guide to the statistical inference tools in Python, we provide code and plots which show how you can illustrate the theory in your classroom and use the tools to solve relevant problems. We have striven as much as possible to come up with examples that are interesting to students and relevant to an international audience, but you might choose to make some changes to the examples to suit your own context and audience.

6.1.2 Statistics Packages in Python

The main Python packages that provide statistics tools are `scipy` and `statsmodels`. Unfortunately, at the time of writing there is no single package that provides all the functionality we need, so we must use a mix of these. These are loaded into the Python environment through the following commands, which appear at the top of each of the interactive notebooks. These commands will only run if you have the packages installed in your environment. You can find instructions for how to do that in the introductory sections of this book.

```
import scipy
import statsmodels.api # appear to need to import the api as well as the library␣
```

(continues on next page)

(continued from previous page)

```
↪itself for the interpreter to find the modules
import statsmodels as sm
```

Using `scipy`'s `stats` modules, one can access the features related to random variables and distributions, e.g., `scipy.stats.norm` for the normal distribution. From there you can randomly sample from the distribution, calculate probabilities with the `.cdf(...)` command, and even get the density function using `.pdf(...)`. Each of these are illustrated later in the text. We also use `scipy.stats` for calculating confidence intervals with the normal distribution, and one sample T-tests.

The `statsmodels` library provides a range of functionality for high level statistics, but here we only make use of a few of its functions. We use the `statsmodels` library for hypothesis tests about a population proportion, for two-sample T-tests, and for linear regression.

For static plots, we are primarily using `seaborn` and `matplotlib`. You can find more information on plotting and statistical graphs using Python earlier in this book.

Some plots and figures in the text are mostly for teaching and illustrating the theory, rather than demonstrating a statistician's actual work process, and you will find that the code for these plots is hidden by default. A number of these plots are interactive, generated using the `plotly` package.

We import the graphics libraries in the following way.

```
import matplotlib.pyplot as plt
from matplotlib import ticker
import seaborn as sns
%matplotlib inline

import plotly.graph_objects as go
import plotly.offline
plotly.offline.init_notebook_mode(connected=True)
```

Finally, we also make use of the `numpy`, `pandas` and `math` libraries, and assume the reader is familiar with these. You can find more information on these libraries in the Introduction to Python sections earlier in the book.

```
# import libraries
# Always run this cell first!
import numpy as np
import pandas as pd
import math
```

```
# import libraries
# Always run this cell first!
import numpy as np
import pandas as pd
import math

import scipy
import statsmodels.api # appear to need to import the api as well as the library␣
↪itself for the interpreter to find the modules
import statsmodels as sm

import matplotlib.pyplot as plt
from matplotlib import ticker
import seaborn as sns
%matplotlib inline
```

(continues on next page)

(continued from previous page)

```
import plotly.graph_objects as go
import plotly.offline
plotly.offline.init_notebook_mode(connected=True) # make plotly work with Jupyter␣
↪Notebook using CDN
```

6.2 Foundations of Statistical Inference

This chapter is all about how to use statistics to make decisions from data, a process known as statistical inference. Because data is usually incomplete and has a lot of variability, this decision-making process inherently contains a lot of uncertainty. But the difficulty shouldn't deter us! The statistical decision-making process is based on a robust theory of probability, and one of the easiest ways to illustrate this theory is using computation.

In this section we take a look at one of the main probability results that allows us to do statistical inference: the Central Limit Theorem. We'll then show how the Central Limit Theorem leads us to the first tool of statistical inference, the confidence interval.

6.2.1 Sampling and the Central Limit Theorem

To get us thinking about statistical decision-making, let's imagine a hypothetical scenario where we are an agent for a football player (that's *soccer* to Americans). As an agent, we want to negotiate a good deal for our client, who plays in the English Premier League. To do this we might need to first try to estimate the average salary for all players in the league.

Unlike a real-world statistical estimate, in this scenario we have access to the entire *population* of player salaries for the 2022-23 Premier League season. This data comes from FBref and is in units of Great British pounds (£, GBP) per week.

Here we will read in the data, which is in the file `EPL_wages_22_23.csv` (click here to download a copy), and display the first 10 rows. These correspond to the 10 highest paid players—you may recognize some of the names!

```
salaries = pd.read_csv("EPL_wages_22_23.csv").drop(columns=["Rk","Notes", "Player-
↪additional","Annual Wages"])
salaries["Weekly Wages (GBP)"] = salaries["Weekly Wages"].str.extract('(\\d+)').
↪astype(int)
salaries.head(n=10)
```

```
              Player   Nation   Pos            Squad  Age  \
0  Cristiano Ronaldo   pt POR    FW   Manchester Utd   37
1    Kevin De Bruyne   be BEL    MF  Manchester City   31
2       David de Gea   es ESP    GK   Manchester Utd   31
3     Erling Haaland   no NOR    FW  Manchester City   22
4       Jadon Sancho  eng ENG  FWMF   Manchester Utd   22
5      Mohamed Salah   eg EGY    FW        Liverpool   30
6     Raphaël Varane   fr FRA    DF   Manchester Utd   29
7    Raheem Sterling  eng ENG  FWMF          Chelsea   27
8           Casemiro   br BRA    MF   Manchester Utd   30
9      Jack Grealish  eng ENG    FW  Manchester City   26

                  Weekly Wages  Weekly Wages (GBP)
0  £ 515000 (€ 583130 $619550)              515000
1  £ 400000 (€ 452916 $481204)              400000
2  £ 375000 (€ 424609 $451129)              375000
3  £ 375000 (€ 424609 $451129)              375000
```

(continues on next page)

(continued from previous page)

```
4  £ 350000 (€ 396302 $421053)              350000
5  £ 350000 (€ 396302 $421053)              350000
6  £ 340000 (€ 384979 $409023)              340000
7  £ 325000 (€ 367994 $390978)              325000
8  £ 300000 (€ 339687 $360903)              300000
9  £ 300000 (€ 339687 $360903)              300000
```

One of the key ideas in statistics is that even if the distribution of the population (in this case, footballers' weekly salaries) is not normal, the distribution of the average salary of a large number of *randomly selected* footballers *does* have an (approximately) normal distribution. Hence, even though any estimation we make from a sample contains some uncertainty, we can quantify that uncertainty and make the best decision possible.

Note: Of course, in real life there are probably a lot more helpful things to calculate than simply the average player salary. For instance, we could look at salaries of players with similar statistics. (Take a look at the *section on linear algebra and clustering* for some ideas.)

We begin by taking a look at the population of all player salaries. Note that this distribution is highly skewed, and far from being normal. In fact, the distribution of player salaries is modeled fairly well with an exponential distribution! Below, we calculate the mean and standard deviation of the weekly wages from this dataset, and compare the histogram of weekly wages to the normal distribution with the same mean and standard deviation as well as the exponential distribution with the same mean.

```
pop_mean = salaries["Weekly Wages (GBP)"].mean()
pop_sd = salaries["Weekly Wages (GBP)"].std()
print(f"Population mean is {pop_mean:.3f}, standard deviation is {pop_sd:.3f}.")
```

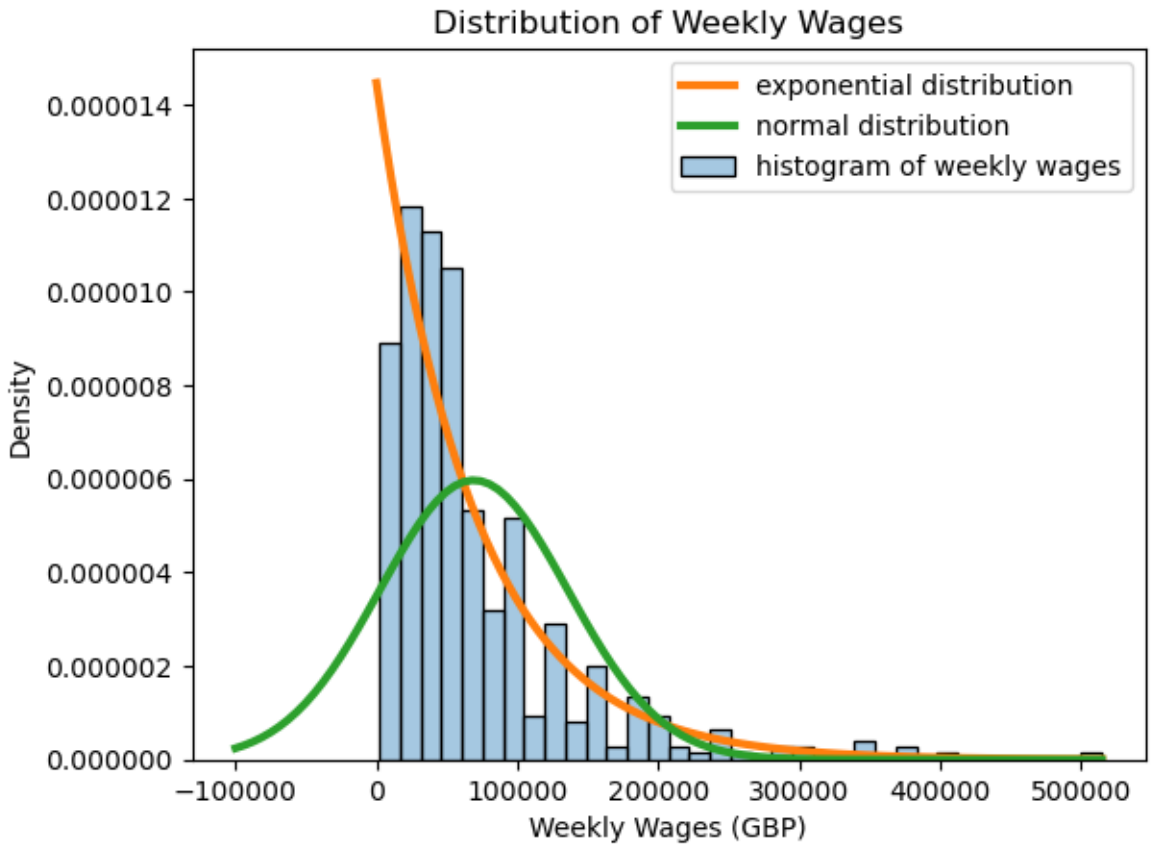
```
Population mean is 69108.187, standard deviation is 66860.742.
```

```
weekly_wages = salaries['Weekly Wages (GBP)']

# use default seaborn palette
palette = iter(sns.color_palette())

# plot histogram
axis = sns.histplot(
    data = weekly_wages,
    stat = 'density',
    color = next(palette), # use next color in palette
    alpha = 0.4,
    label = 'histogram of weekly wages'
)

# plot exponential distribution
x = np.linspace(0, max(weekly_wages), 100)
axis.plot(
    x,
    scipy.stats.expon.pdf(x, scale=np.mean(weekly_wages)),
    color = next(palette), # use next color in palette
    linewidth = 3,
    label = 'exponential distribution'
)

# plot normal distribution
x = np.linspace(-100000, max(weekly_wages), 100)
axis.plot(
    x,
```

(continues on next page)

(continued from previous page)

```
    scipy.stats.norm.pdf(x, loc=np.mean(weekly_wages), scale=np.std(weekly_wages)),
    color = next(palette), # use next color in palette
    linewidth = 3,
    label = 'normal distribution'
)

axis.ticklabel_format(style='plain', axis='y')

axis.set_title("Distribution of Weekly Wages")
axis.legend();
```



In our hypothetical scenario, let's imagine that we didn't have access to *all* of the data, but only knew the salaries for 35 random players. We could take the average salary of those 35 players, but would it be close to the real average salary of *all* the players?

The following code will take a random sample of 35 players, and compute the (sample) average of the salaries of those players. Is it close to the actual salary?

```
my_sample = salaries.sample(n=35)

sample_mean = my_sample["Weekly Wages (GBP)"].mean()
sample_sd = my_sample["Weekly Wages (GBP)"].std()

print(f"Sample mean is {sample_mean:.3f}.")
```

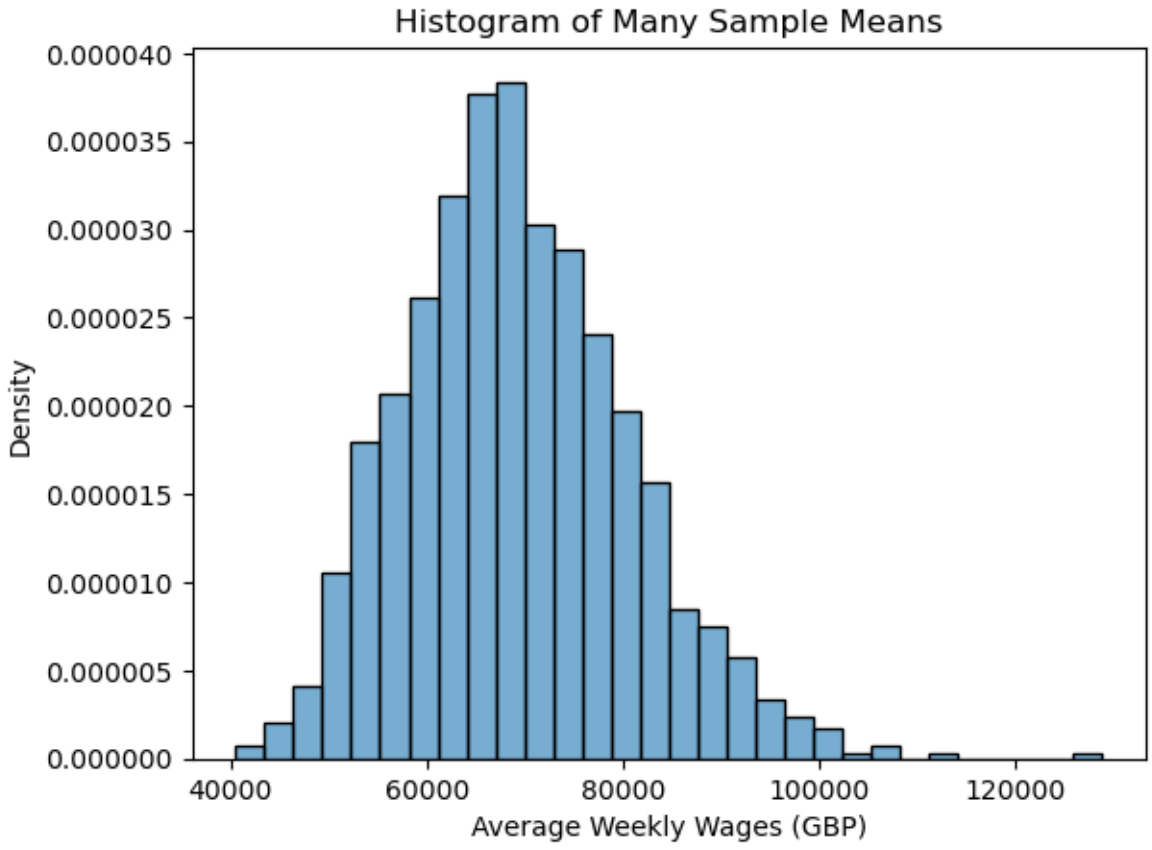
```
Sample mean is 76742.857.
```

It's important to note that this sample mean is just one possible value. If we had taken a different random sample of 35 players, we would get a different sample average. You can see this for yourself if you run the code several times. This is why we interpret the sample mean as a random variable.

As an example, if we happened to randomly selected highly-paid players such as Cristiano Ronaldo, we would expect our sample average to be higher than the population average. If we randomly selected a lot of young players or reserves, we would expect our sample average to be lower. But how much lower or higher is possible? How much lower or higher should we expect? We can answer this question by understanding sampling distributions. In other words, if the sample mean is a random variable, what is its distribution?

Because we have access to the entire population of data, we can explore this idea through simulation. Instead of taking just one sample mean from a sample of size 35, we can take many of them and look at the possible values. The following code generates 1000 different random samples of 35 players, and records the average salary of each sample. We then make a relative frequency histogram to show which average salaries occurred most often across the 1000 random samples.

```
# calculate the means of 1000 random samples of size 35
many_sample_means = [salaries.sample(n=35)["Weekly Wages (GBP)"].mean() for i in␣
 ↪range(1000)]

# plot histogram of many_sample_means
axis = sns.histplot(
    data = many_sample_means,
    stat = 'density',
    alpha = 0.6,
)
axis.ticklabel_format(style='plain', axis='y')
axis.set_xlabel("Average Weekly Wages (GBP)")
axis.set_title("Histogram of Many Sample Means");
```



Now let's compare this to the theory. What does the Central Limit Theorem tell us?

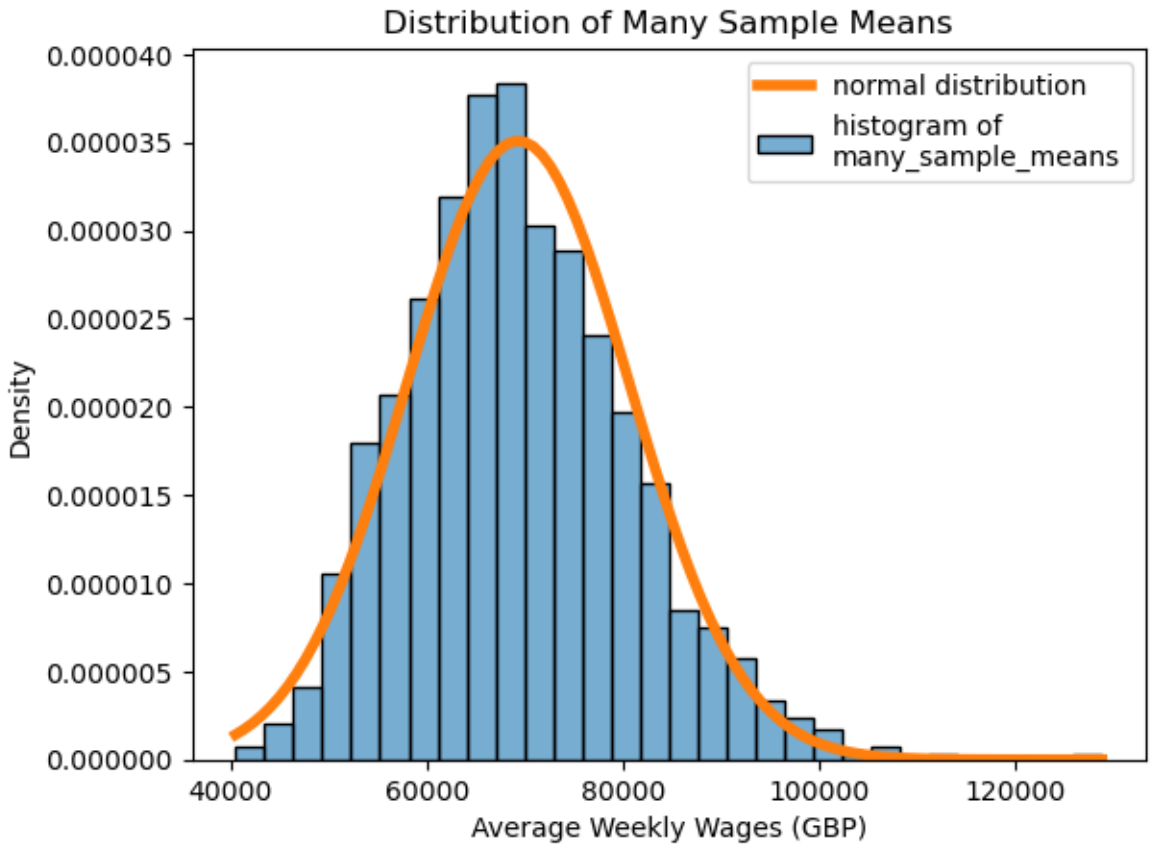
If the sample size is "large enough", then the sample mean, considered as a random variable, is approximately normally distributed with the same mean as the population and with standard deviation equal to the population standard deviation divided by the sample size. In mathematical language,

$$\overline{X} \sim N\left(\mu, \frac{\sigma}{\sqrt{n}}\right), \tag{6.1}$$

where $\overline{X}$ is the sample mean, μ is the population mean, σ is the population standard deviation, and n is the sample size.

In our case, $n = 35$, and the population mean and standard deviation were calculated above. Let's replot our histogram of `many_sample_means`, along with the normal distribution with the same mean and standard deviation, and see how it matches up with the theory.

```
# Same histogram as above, with normal distribution overlay

# use default seaborn palette
palette = iter(sns.color_palette())

# plot histogram
histcolor = next(palette) # use next color in palette
axis = sns.histplot(
    data = many_sample_means,
    stat = 'density',
    color = histcolor,
    alpha = 0.6,
    label = 'histogram of\nmany_sample_means'
)

# plot normal distribution
x = np.linspace(min(many_sample_means), max(many_sample_means), 100)
axis.plot(
    x,
    scipy.stats.norm.pdf(x, loc=np.mean(many_sample_means), scale=np.std(many_sample_
 ↪means)),
    color = next(palette), # use next color in palette
    linewidth = 4,
    label = 'normal distribution'
)

axis.ticklabel_format(style='plain', axis='y')
axis.set_xlabel("Average Weekly Wages (GBP)")
axis.set_title("Distribution of Many Sample Means")
axis.legend();
```



We now see that even though the average weekly salary of 35 players is variable, and changes based on which 35 players we get, overall we have a good idea of which values we're likely to get!

All of this depends on the sample size being "large enough", which leaves us with a big question: how large is large enough?

You can play around with the interactive widget below to try different sample sizes and see how well the theoretical normal distribution matches up with the sampling distribution. Given a sample size, the widget plots the average of 10000 random samples of that size from the salary data, overlaid with the normal distribution with the same mean and standard deviation.

Drag the slider to select different sample sizes, and click the labels in the legend to hide or show different parts of the graph. Notice how the histogram of the averages fits the normal distribution more closely as the sample size increases.

Note

Please visit the web version of this section to engage with this interactive resource.

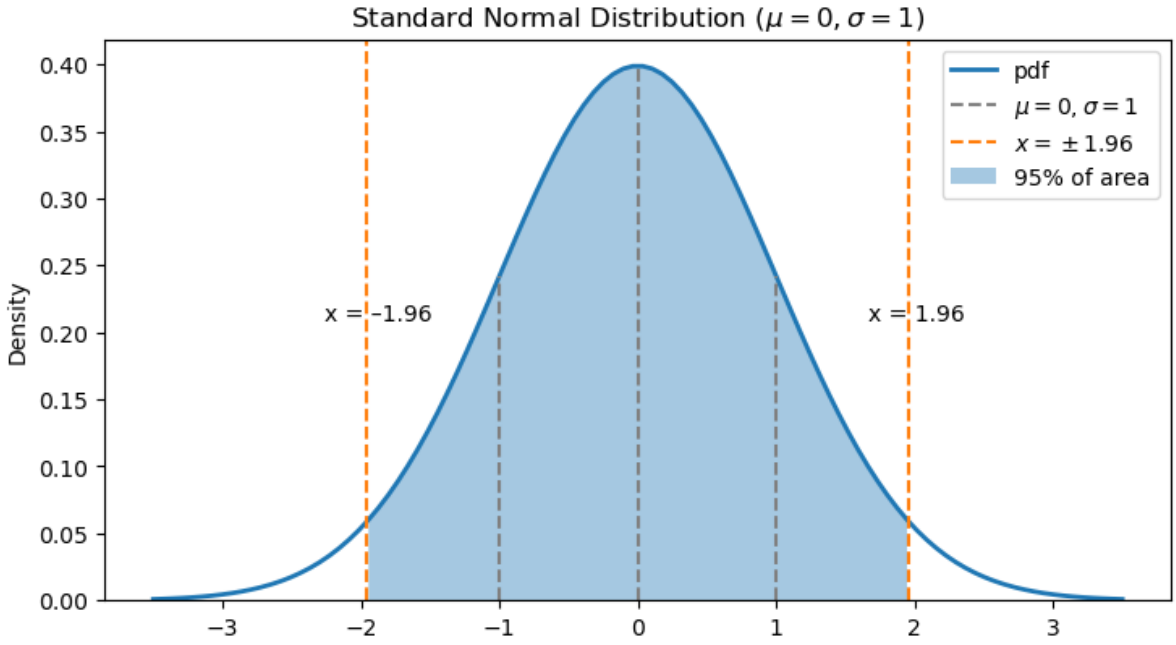
Statisticians will usually say that a sample size of 30 is large enough. Based on the widget above, would you agree with that? Do the distributions look approximately normal at that sample size?

6.2.2 Confidence Intervals

In real life, the situation is much different. We will typically only have one sample, and hence one sample mean. We now know that if our sample size n is large enough, we are likely to be close to the true population mean. But how close are we? And can we quantify what "likely" means in this case?

The Central Limit Theorem (CLT) tells us where $\overline{X}$ is likely to be in relation to μ. Since $\overline{X} \sim N(\mu, \frac{\sigma}{\sqrt{n}})$, we know that 95% of the time $\overline{X}$ will fall within a distance of about 2 standard deviations from μ (technically, the distance is more like 1.96). In other words, 95% of the area under the density function for $\overline{X}$ is between $\mu - 1.96\frac{\sigma}{\sqrt{n}}$ and $\mu + 1.96\frac{\sigma}{\sqrt{n}}$.

The plot below illustrates what 95% of the area under the standard normal density function looks like.



Now we know where $\overline{X}$ is likely to fall in relation to μ. We can express what we see in the graph as a probability:

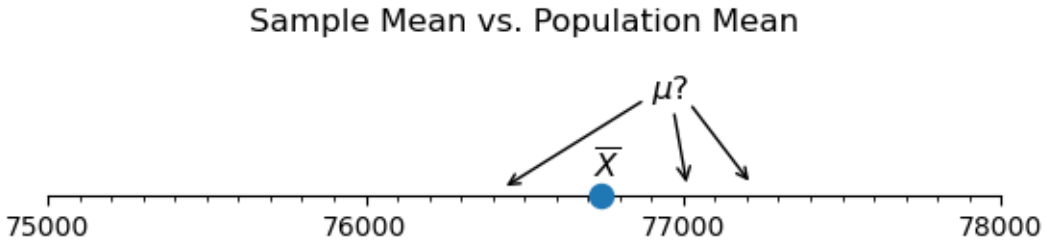
$$P\left(\mu - 1.96\frac{\sigma}{\sqrt{n}} \leq \overline{X} \leq \mu + 1.96\frac{\sigma}{\sqrt{n}}\right) = .95. \tag{6.2}$$

With a little bit of algebraic manipulation, we can write this as an interval for μ:

$$P\left(\overline{X} - 1.96\frac{\sigma}{\sqrt{n}} \leq \mu \leq \overline{X} + 1.96\frac{\sigma}{\sqrt{n}}\right) = .95. \tag{6.3}$$

In the real world, we'll have a value of $\overline{X}$, and we want to know where μ is. This second interval gives us exactly this! Look at what's on the upper and lower side of the interval: (almost) all things we can calculate from the data, like $\overline{X}$ and n (we'll discuss how to handle the σ in a moment). This gives an interval for μ, the population mean that we can't calculate but want to know.

Let's return to our hypothetical case. We have one sample of $n = 35$ players, and one observed sample mean. Since, hypothetically, we don't know the population mean, can we give an interval where we think it is?



We know from above that there is a 95% probability that for a random value of $\overline{X}$, the interval $\left(\overline{X} - 1.96\frac{\sigma}{\sqrt{n}}, \overline{X} + 1.96\frac{\sigma}{\sqrt{n}}\right)$ will contain μ. Let's compute it!

We have access to the sample mean $\overline{X}$:

```
print(f"Sample mean is {sample_mean:.3f}.")
```

```
Sample mean is 76742.857.
```

We don't know the population standard deviation σ, but we might expect that the sample standard deviation S is sufficiently close. If you don't believe that, we could also use a T-distribution, introduced in the *next section*. But in this case, our sample size is $n = 35$, which is large enough to use S in place of σ.

```
print(f"Sample standard deviation is {sample_sd:.3f}.")
```

```
Sample standard deviation is 90566.304.
```

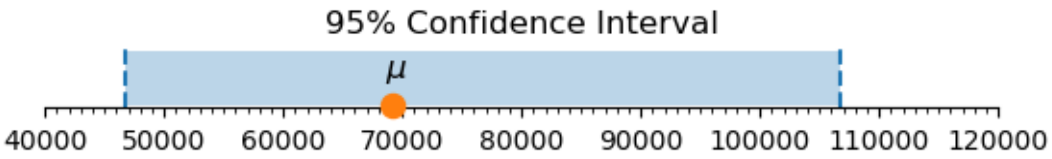
Let's put it all together based on our formula above:

```
lower_ci = sample_mean - 1.96 * sample_sd / np.sqrt(35)
upper_ci = sample_mean + 1.96 * sample_sd / np.sqrt(35)

print(f"95% confidence interval is ({lower_ci:.3f}, {upper_ci:.3f}).")
```

```
95% confidence interval is (46738.198, 106747.516).
```

We are 95% confident that the average weekly salary for premier league players falls in the interval we calculated.

In our case, we actually have access to the real population parameter. Does our calculated interval actually contain the true population mean μ? Let's visualize it:



6.2.3 Confidence Intervals in Python Libraries

We don't always have to calculate confidence intervals by hand. There are many Python libraries that will do this for you! One way is to use the `scipy.stats` library. We'll also find in other sections that some of the statistical inference code we run will automatically provide a confidence interval. The previous demonstrations will hopefully be helpful for you to know what's going on beneath the surface of the code and to give you a bit more *confidence* in the computation.

```
confidence_level = 0.95
(lower_ci, upper_ci) = scipy.stats.norm.interval(
    confidence_level,
    loc=sample_mean,
    scale=sample_sd/np.sqrt(35)
)
print(f"95% confidence interval is ({lower_ci:.3f}, {upper_ci:.3f}).")
```

```
95% confidence interval is (46738.750, 106746.965).
```

Of course, 95% is not the only value we could use for the confidence level, though it is the most common. If we wanted to be more certain, we could increase that value. If precision was less important, we could decrease it. It all depends on the context in which you are working!

In later sections, we will primarily compute confidence intervals for means using the T distribution, rather than the normal distribution. The difference between the intervals is discussed in those places.

6.2.4 Why Is It Called a Confidence Interval?

The number 95% is a probability, but it is not the probability that μ, the population mean, is in the interval. The population mean doesn't change, and hence it is either in the given interval or not—no probabilities involved! So what does the 95% confidence mean?

The probability 95% concerns the interval itself, which depends on the data that we used. A different sample would give us different data, and hence a different interval. The "confidence" in "95% confidence" refers to the sampling process. Another way to say this is that 95% of all the possible confidence intervals we could construct in this way contain the true population mean.

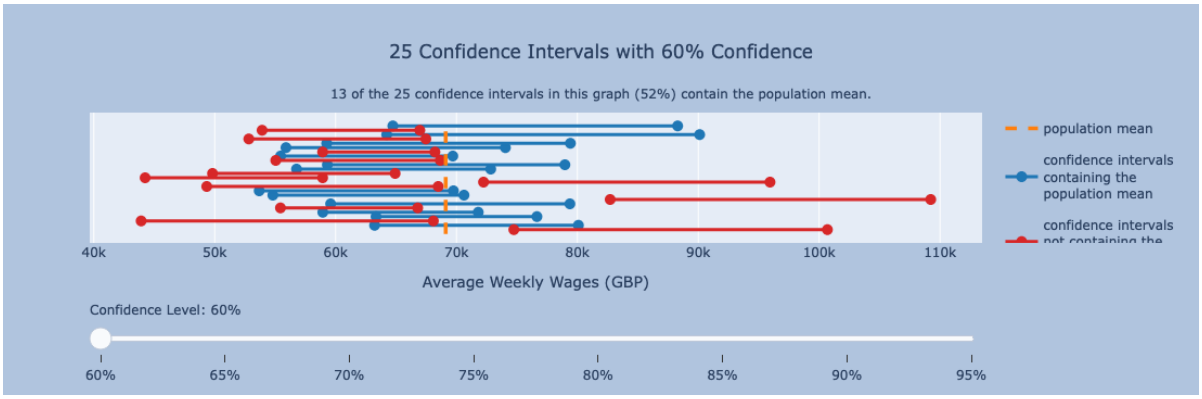
Since we have access to an entire population of footballers' salaries, we can take many samples and construct many sample means and confidence intervals. If we did this, how many would we expect to have the true average weekly salary (the population mean)?

You can play around with the interactive widget below to explore this idea. Given a confidence level, the widget takes 25 random samples of size 35 from the salary data and plots a confidence interval for each one. (This widget is inspired by the OpenIntro lab on confidence intervals, which is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.)

Drag the slider to select different confidence levels, and click the labels in the legend to hide or show different parts of the graph. Notice how different confidence levels result in different numbers of confidence intervals containing the population mean. (These numbers might not exactly match the confidence level, but that's because we're only constructing 25 confidence intervals, which is far short of the *total* number of possible confidence intervals we could construct.)

Note

Please visit the web version of this section to engage with this interactive resource.

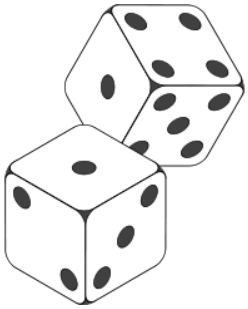


6.2.5 Exercise 1

1. Use the widget above to construct 25 different 90% confidence intervals for the average weekly wages of Premier League players. How many of the 25 contain the true population mean? Is this the number you expect? Would it be possible for exactly 90% of the intervals to contain the true population mean?
2. Choose your own confidence level and sample size, and construct a single confidence interval for the average weekly wages of a Premier League player. Describe your interval in a complete sentence.

6.2.6 The Central Limit Theorem for Discrete Numerical Random Variables

Finally, we want to note that this theory holds for any numerical distribution, even one that is discrete! In the widget below, we demonstrate the CLT for discrete distributions by approximating the sample average of a number of dice rolls.



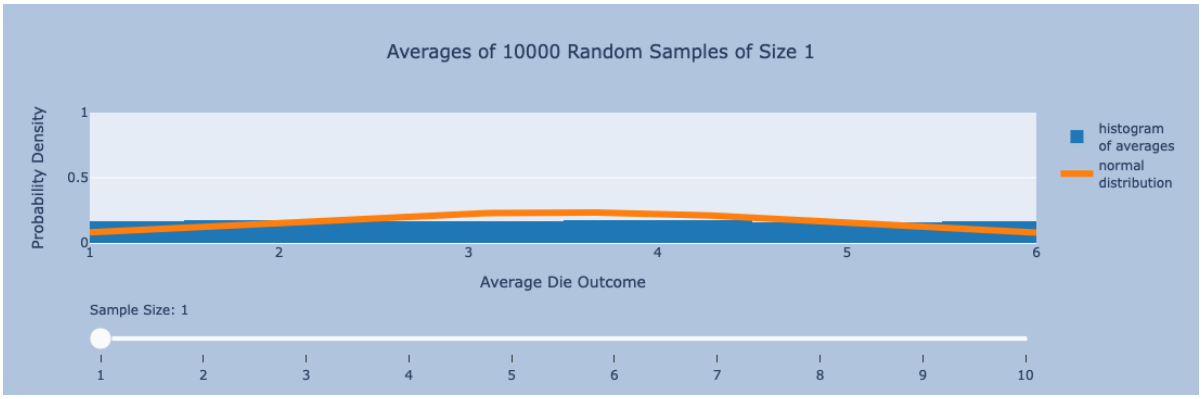
Dice icon by Steaphan Greene, licensed under CC BY-SA 3.0.

Given a sample size, the widget simulates rolling a die that many times and then takes the average of the rolls. It does this 10000 times and then plots the histogram of the resulting averages, overlaid with the normal distribution with the same mean and standard deviation.

Drag the slider to select different sample sizes, and click the labels in the legend to hide or show different parts of the graph. Notice how the histogram of the averages fits the normal distribution more closely as the sample size increases.

Note

Please visit the web version of this section to engage with this interactive resource.



6.2.7 Confidence Intervals for Categorical Data/Proportions

We can also calculate confidence intervals for population proportions and differences of proportions. We'll explore that in the *next section* alongside hypothesis testing!

```
# import libraries
# Always run this cell first!
import numpy as np
import pandas as pd

import scipy
import statsmodels.api # appear to need to import the api as well as the library itself for the interpreter to find the modules
import statsmodels as sm

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import plotly.graph_objects as go
import plotly.offline
plotly.offline.init_notebook_mode(connected=True) # make plotly work with Jupyter Notebook using CDN
```

6.3 Hypothesis Testing for Categorical Data

6.3.1 Statistical Decision-Making: The Hypothesis Test

In the *last section*, we saw how probability theory can help us to estimate how close a sample average is to the true average of a population using a confidence interval.

In this section, we will look at another important tool in the statistician's repertoire, the statistical test of hypotheses. Hypothesis testing works much like the scientific method: we make a hypothesis about how the world works, construct an experiment to test it, gather data from the experiment, and then evaluate how closely that data fits the hypothesis.

Let's look at this process in the context of an example.

The Wellcome Global Monitor Survey

The Wellcome Global Monitor survey, which is part of the World Gallup Poll, asks people around the world about their views on topics related to science and health. The most recent data, as of writing, is from 2018 and consists of survey answers from over 140,000 people in 144 different countries, collected in person or by telephone. The data is available under a Creative Commons Attribution 4.0 International License.

One of our goals in this chapter is to help you understand the statistical decision-making process, and, through learning more, to have more confidence about how scientists move forward in their understanding. As a statistician and educator, I'm very interested in the level of trust people have in scientists and the scientific process.

Hence, we are going to look today at the Wellcome Global Monitor's Trust in Scientists Index, which measures how much each respondent trusts scientists based on their answers to five of the questions in the survey:

- "How much do you trust scientists in your country?"
- "In general, how much do you trust scientists to find out accurate information about the world?"
- "How much do you trust scientists working in colleges/universities in this country to do their work with the intention of benefiting the public?"
- "How much do you trust scientists working in colleges/universities in this country to be open and honest about who is paying for their work?"
- "How much do you trust scientists working for companies in this country to do their work with the intention of benefiting the public?"

The answers are combined to rank each respondent's level of trust in scientists from "High Trust" to "Low Trust". Respondents that did not answer enough of the questions are recorded as "No Score".

The survey data we have is contained in a file named `wellcome_global_monitor_2018.csv` (click here to download a copy). This file contains a subset of the attributes from the original dataset, replacing numerical codes with their textual equivalents. The column labeled `Trust_Index` (originally named "WGM_Indexr") contains the Trust in Scientists Index.

```
wgm = pd.read_csv('wellcome_global_monitor_2018.csv')
wgm.sample(5).head() # display 5 randomly sampled rows
```

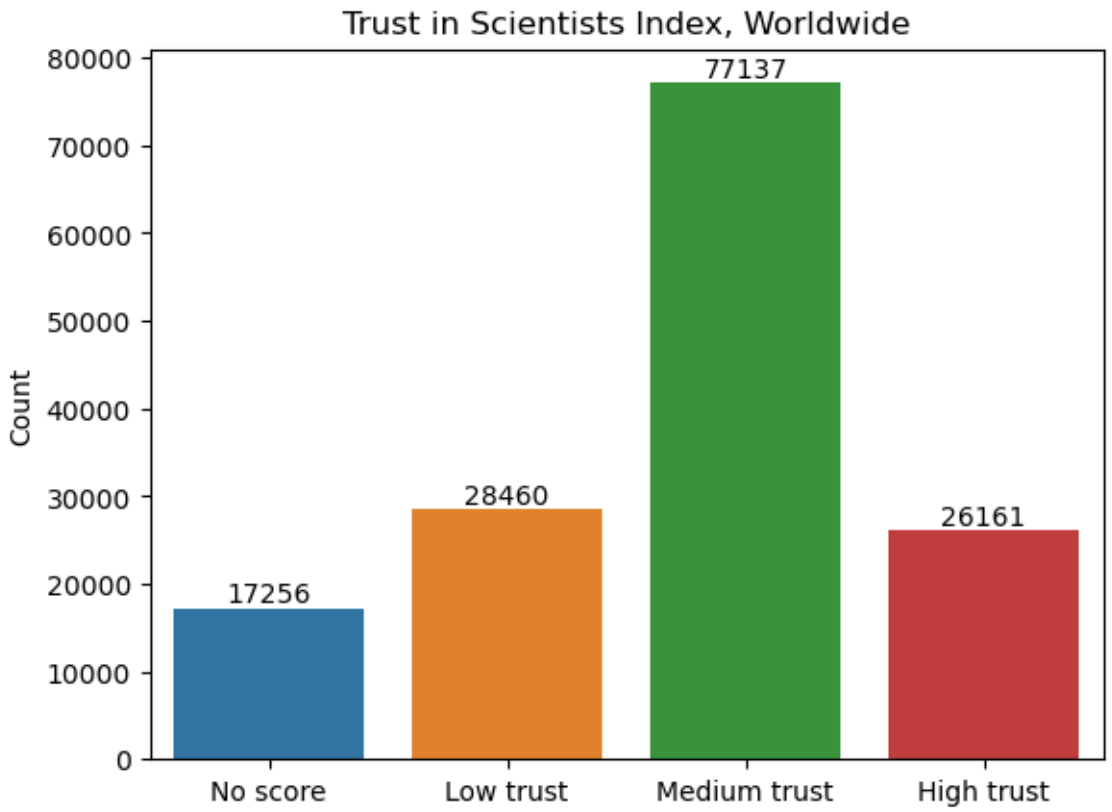
```
                        Country  Gender   Trust_Index
95552                  Colombia  Female      No score
4540                Saudi Arabia  Female      No score
85209                  Argentina    Male  Medium trust
144830                   Uruguay  Female  Medium trust
41201   Palestinian Territories    Male  Medium trust
```

Let's take a deeper look at the Trust in Scientists Index and see what levels of trust were recorded.

```
wgm["Trust_Index"].value_counts()
```

```
Trust_Index
Medium trust    77137
Low trust       28460
High trust      26161
No score        17256
Name: count, dtype: int64
```

```
# create dataframe from counts of "Trust_Index" values
levels = wgm["Trust_Index"].value_counts().reset_index(name="Count").rename(columns={
 ↪"index": "Trust_Index"})

# create bar plot
ax = sns.barplot(
    data = levels,
    x = "Trust_Index",
    y = "Count",
    order = ["No score", "Low trust", "Medium trust", "High trust"]
)

# label each bar with the corresponding value
for container in ax.containers:
    ax.bar_label(container)

# set axis labels and title
ax.set_xlabel("")
ax.set_title("Trust in Scientists Index, Worldwide");
```



In what follows, we are going to narrow our analysis to focus in on one country. In this case, we will look at the United States. If you're interested in a different country, later on you'll get a chance to repeat this analysis for a different country in the exercises. Or, if you're using the interactive notebook, you can change the code to use a different country!

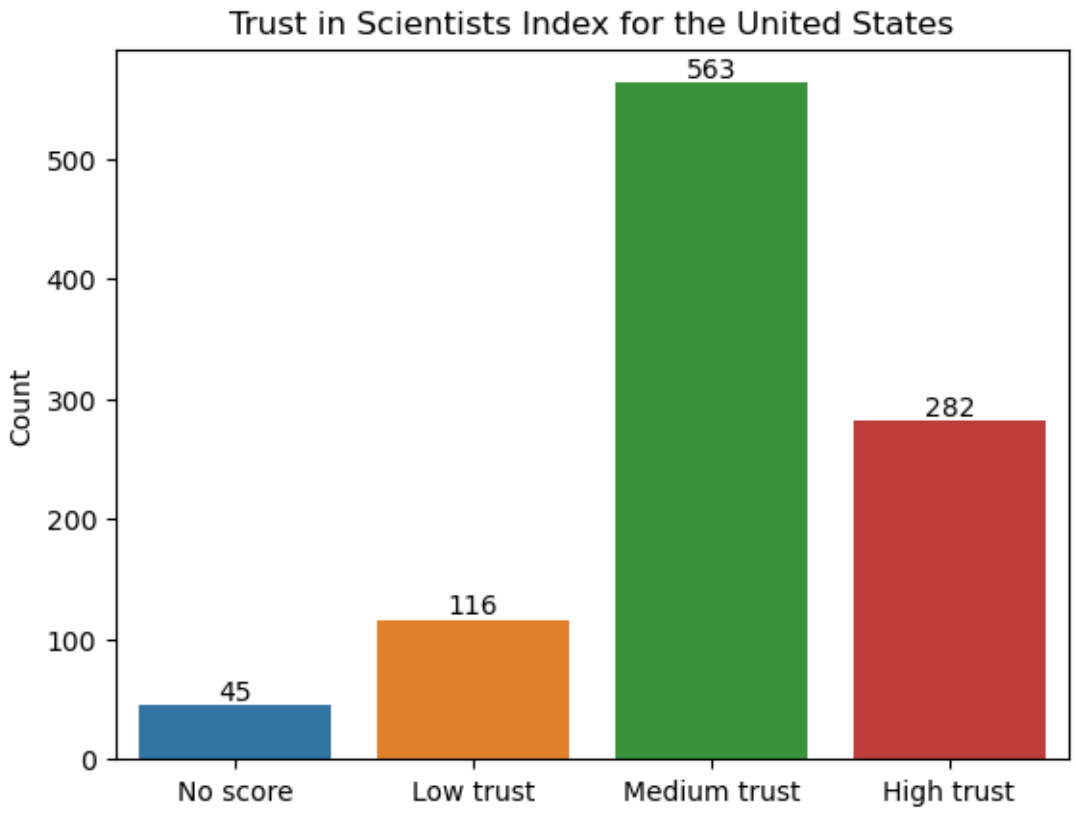
```
wgm_usa = wgm[wgm['Country'] == "United States"]
```

Here's what the levels of trust for the sample from the United States look like:

```
wgm_usa["Trust_Index"].value_counts()
```

```
Trust_Index
Medium trust    563
High trust      282
Low trust       116
No score         45
Name: count, dtype: int64
```

```
# create dataframe from counts of "Trust_Index" values
levels = wgm_usa["Trust_Index"].value_counts().reset_index(name="Count").
 ↪rename(columns={"index": "Trust_Index"})

# create bar plot
ax = sns.barplot(
    data = levels,
    x = "Trust_Index",
    y = "Count",
    order = ["No score", "Low trust", "Medium trust", "High trust"]
)

# label each bar with the corresponding value
for container in ax.containers:
    ax.bar_label(container)

# set axis labels and title
ax.set_xlabel("")
ax.set_title("Trust in Scientists Index for the United States");
```



Making a Hypothesis

The first step in a hypothesis test is to determine the hypotheses that we are comparing. We are comparing two opposing views of the world: either it works like this, or it doesn't. These views are best understood in terms of *models*—a framework for understanding how the world works.

For example, perhaps I think that the proportion of residents in the United States who have a high trust in scientists is large, say over 0.25. In this situation, the two models are statistical: we are hypothesizing about the proportion from a population. Either the proportion of people in our chosen country who have high trust in scientists is 0.25, or it's more than that. In mathematical terms, we'll write this as

$$H_0 : p = 0.25, H_1 : p > 0.25. \tag{6.4}$$

In other words, our null hypothesis, H_0, is that the proportion is 0.25. Our alternative hypothesis, H_1, is that the proportion is greater than 0.25. Remember that the null hypothesis should always be in the form "population parameter equals something", even though in this case the null hypothesis will also include cases when the proportion is *below* 0.25.

We can make hypotheses about other parameters, such as mean or standard deviation, but we'll start here by working with proportions.

The statistical hypothesis test is the method for making a decision between these two models. We gather data and see how well the data fits with what we would expect based on the null hypothesis.

How Much Evidence Do We Need?

After we have determined the hypotheses, the next step is to determine our threshold for making a decision. How much evidence do we need to reject our null hypothesis in favor of the alternative hypothesis? Another way to think about this is, how much risk are we willing to take to be wrong?

In this case, there are actually *two* ways to make the wrong conclusion: we could reject our null hypothesis when it is actually true (called a **Type I error**), or we could fail to reject the null hypothesis when it is actually false (a **Type II error**). We won't go any deeper into Type II error at the moment, but it is just as important to consider!

The "risk level" or "evidence threshold" associated with the first type of wrong conclusion is denoted by the Greek letter α (alpha). For any given example, the value should be chosen based on both the context of the problem and the risk of making a Type I error that you are willing to take. Typical values for α in scientific literature are 0.1, 0.05, and 0.01, but these values are somewhat arbitrary.

For instance, imagine you are trying to determine if a new medical treatment is more effective than a previous treatment. Suppose the new treatment has more potential for harmful side-effects than the previous treatment. In this case, you would want to be very sure the new treatment is better before you start using it widely, so you would want to set your α-value to something smaller to make sure that the risk of making a Type I error is not too large.

```
alpha = 0.05
```

The Test Statistic: Using Data to Decide

After determining the value of α, the next step of a real-world study would be to gather the data we need and make a decision. In the statistics classroom, you typically have the data or summary statistics already available to you, as we do in this case.

We are asking questions about the proportion of people in the United States who have high trust in scientists. It would be impossible to get an answer from every single person in the entire country! However, the Wellcome Global Monitor data gives us answers from a large representative sample of people in the USA. So we can use this data to get a statistically sound idea of what the true proportion in the entire population is.

Let's compare the proportion from the sample to what the null hypothesis says it should be. If the sample proportion is close to what the null hypothesis says it should be, that would mean that the null hypothesis is probably right. On the other hand, if it is different, perhaps the null hypothesis is wrong!

We can compute the sample proportion of people with high trust in scientists by taking the total number of people in the sample with a high trust score and dividing it by the total sample size:

$$\hat{p} = \frac{\text{num. of respondents with high trust in scientists}}{\text{num. of respondents}}. \tag{6.5}$$

```
# number of successes (i.e., people with high trust in scientists)
count = len(wgm_usa[wgm_usa["Trust_Index"] == "High trust"])

# sample size (i.e., the total number of observations)
nobs = len(wgm_usa["Trust_Index"])

# compute sample proportion
p_hat =  count / nobs
print(f"The sample proportion is {p_hat:.3f}. There are {nobs} total observations.")
```

```
The sample proportion is 0.280. There are 1006 total observations.
```

This is clearly more than 0.25, the value in our null hypothesis! But remember that the sample proportion is variable, so it won't be *exactly* the same as the population proportion. So is this evidence for the alternative hypothesis?

We could ask a better question: is the sample proportion of 0.28 a lot different from our null hypothesis or just a little bit different? We can answer this using probability theory. Just like in the *last section*, the theoretical foundation for the hypothesis test is the Central Limit Theorem. Remember that if the sample size n is large enough, sample proportions have a normal distribution, where the mean is the population proportion p, and the standard deviation is the population standard deviation $\sqrt{p(1-p)}$ divided by $\sqrt{n}$:

$$\hat{p} \sim N\left(p, \sqrt{\frac{p(1-p)}{n}}\right).$$

Above we can see that $n = 1006$. If we assume the null hypothesis is true, then we also know what to fill in for p:

$$\hat{p} \sim N\left(.25, \sqrt{\frac{.25(1-.25)}{1006}}\right).$$

We call this the **null distribution**, since it is the distribution of the sample statistic under the assumption that the null hypothesis is true. (This is another reason why we always state the null hypothesis in terms of "population parameter equals some value".)

Back to the question: is the difference between the observed value of $\hat{p}$ and the value that the null hypothesis says it should be a big difference or small difference? One way to answer that question would be to **standardize** by taking the difference between the sample mean and the (hypothesized) population mean and dividing by the standard deviation:

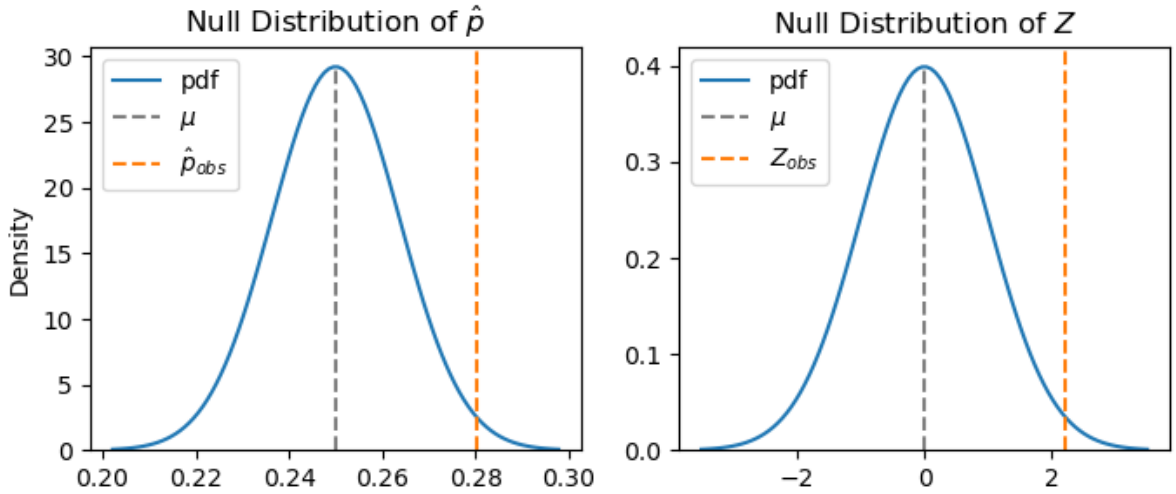
$$Z = \frac{\hat{p} - .25}{\sqrt{\frac{.25(1-.25)}{1006}}} \sim N(0,1) \tag{6.6}$$

This gives us the standardized test statistic Z, which has a standard normal distribution ($\mu = 0, \sigma = 1$)—again assuming that the null hypothesis is true.

```
# standardize it
z_obs = ( p_hat - 0.25 ) / np.sqrt( 0.25*(1-0.25) / nobs )
print(f"The observed value of the test statistic Z is {z_obs:.3f}.")
```

```
The observed value of the test statistic Z is 2.221.
```

What's the difference between the distribution of $\hat{p}$ and the distribution of Z? Not much, as you see below. However, note that the scales are different. Standardizing removes the original units and puts everything into units of standard deviations, enabling you to compare statistics that were originally in different units.



This changes our difference to units of standard deviations. In other words, the sample proportion is 2.221 standard deviations above the mean value of 0.

Think about it another way: if the null hypothesis is true, then most of the time $\hat{p}$ will be close to $p = 0.25$, and hence Z will be close to 0. But $\hat{p}$ varies, and hence Z is variable. So how likely is it that Z would be 2.221 standard deviations from what the null hypothesis says it should be most of the time? And how far away does it need to be before we would start to doubt the null hypothesis?

Making a Decision

Now that we have a hypothesis and have calculated a test statistic to try to evaluate the hypothesis, we are ready to make a conclusion. Under the null hypothesis, we know the distribution of the sample statistic $\hat{p}$. However, we have just one observation of the sample proportion.

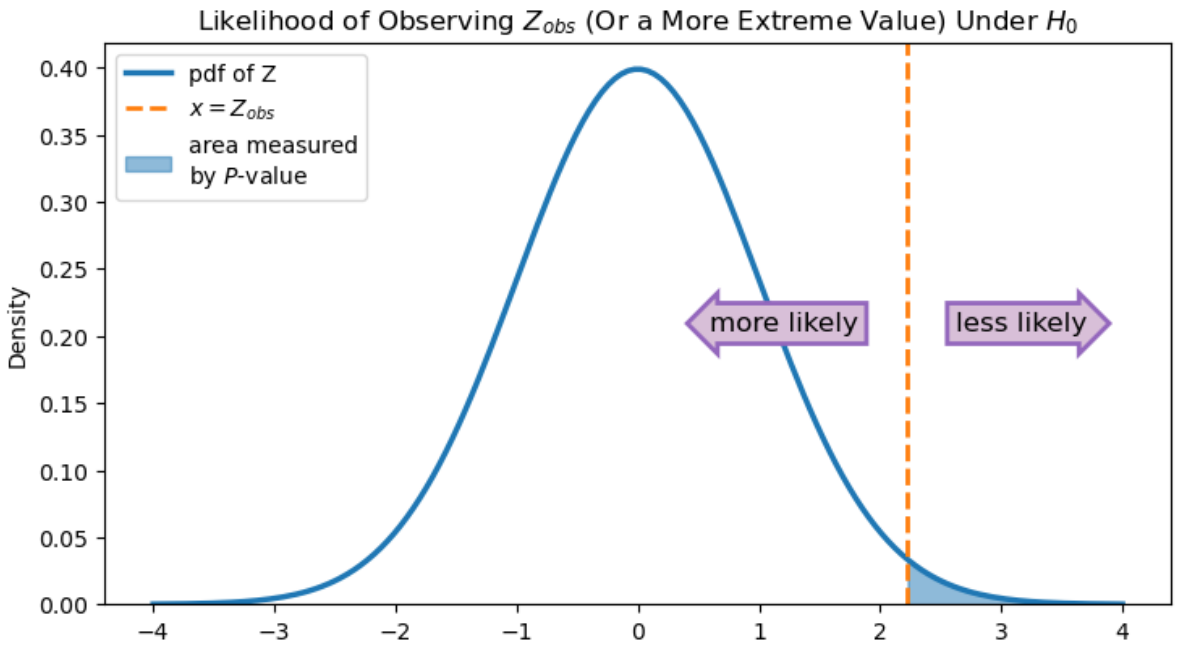
So we ask, if the null hypothesis is true and the population proportion really is 0.25, how likely is it that we would observe the value of 0.28 or something even further away from 0.25? Or translated to Z-space, how likely is it to get a test statistic that is 2.221 or larger?

We can use `scipy.stats` to compute directly compute this probability from a $N(0,1)$ distribution using the CDF function. We subtract the probability from 1 since the CDF always gives the probability to the *left* of a number, and we want the probability that Z is *larger* or to the right of 2.221.

```
1 - scipy.stats.norm(0,1).cdf(z_obs)
```

```
0.013183855477092332
```

This value is called the P-value, and it measures the probability of observing a value at least as much greater than 0.25 as the value we actually observed is (assuming the null hypothesis is true). As the figure below shows, the P-value is the area under the probability density function of Z that is to the right of Z_{obs}. The smaller this area is, the less likely it is that, if the null hypothesis is indeed true and we took additional samples from our population, we would observe values of Z that are greater than or equal to Z_{obs}.



In some sense, the P-value quantifies the evidence you have *against* the null hypothesis. If the P-value is small (i.e., the blue area under the curve is small), then that means that if the null hypothesis is really true, it is unlikely that you would get the value of $\hat{p}$ that you did. In other words, a small P-value indicates that the data is in disagreement with the statistical model given by the null hypothesis.

Based on the P-value, we can make our decision. A P-value is "small enough" if it is below the threshold of α that we set previously. If the P-value is less than α, then our test statistic is incompatible with the null hypothesis, and we will **reject the null hypothesis**. But if the P-value is greater than or equal to α, then our data is compatible with the null hypothesis, and we will **fail to reject the null hypothesis**.

In this case, we have chosen $\alpha = 0.05$, and our P-value is 0.013. Since $0.013 < 0.05$, we will reject the null hypothesis.

Note: Make sure you understand the difference between the population proportion p and the P-value—they are very different! We chose to capitalize the P in P-value to make this distinction more clear, but this is not the way you will find it in every book.

In Python, we can put this all together using the `statsmodels` library, which we imported at the top of the notebook and gave the alias `sm`.

A test for proportions using `statsmodels` is performed below by calling `sm.stats.proportion.proportions_ztest(...)`. The name `proportions_ztest` indicates that we will be using the standard normal distribution to model the test statistic.

The arguments required by the test are all of the things we calculated above: the number of people with high trust in scientists (`count`), the sample size (`nobs`), and the hypothesized value of the proportion (`p0`). Because the alternative hypothesis is that $p > 0.25$, we add `alternative = "larger"`. Other possibilities for the alternative hypothesis are `"smaller"` or `"two-sided"`. The code returns the test statistic and the P-value associated with the hypothesis test, which we store in the variables `stat` and `pval`, respectively.

```
p0 = 0.25 # proportion under null hypothesis
(stat, pval) = sm.stats.proportion.proportions_ztest(
    count = len(wgm_usa[wgm_usa["Trust_Index"] == "High trust"]), # number of␣
↪successes
    nobs = len(wgm_usa["Trust_Index"]), # number of observations
    value = p0,
```

(continues on next page)

(continued from previous page)

```
    alternative = "larger",
    prop_var = p0, # use proportion under null hypothesis to calculate variance for␣
 ↪test statistic
)
print(f"Z-test for proportion: test statistic is {stat:.3f}, P-value is {pval:.3f}.")
```

```
Z-test for proportion: test statistic is 2.221, P-value is 0.013.
```

As you can see, `statsmodels` gives us the same test statistic and P-value that we found earlier. As before, since $0.013 < 0.05$, we reject the null hypothesis. Note that the output of the code doesn't tell you what conclusion to make. Remember again that α, the threshold for making your conclusion, is context-dependent, and the P-value is just one part of making a decision.

Confidence Intervals for Proportions

Now that we've made a decision to reject the null hypothesis, we might want to estimate the actual proportion of people in the USA with high trust in scientists. We can do this using a confidence interval. Confidence intervals for population proportions have a similar theory to the confidence intervals for population means we discussed in the *previous section*, so we won't discuss it here.

We will use a `statsmodels` function called `proportion_confint` to calculate the confidence interval.

```
# one-sample proportion confidence interval (for a binomial proportion)
confidence_level = 0.95
(lower_ci, upper_ci) = sm.stats.proportion.proportion_confint(
    count = len(wgm_usa[wgm_usa["Trust_Index"] == "High trust"]), # number of␣
 ↪successes
    nobs = len(wgm_usa["Trust_Index"]), # number of observations
    alpha = 1-confidence_level,
    method='normal',
)
print(f'{int(confidence_level*100)}% confidence interval is ({lower_ci:.3f}, {upper_
 ↪ci:.3f}).')
```

```
95% confidence interval is (0.253, 0.308).
```

So we have 95% confidence that the actual proportion of USA residents with high trust in scientists is between 25.3% and 30.8%. Notice that we choose our desired confidence level to be 95%, but you could easily adjust this for other cases.

6.3.2 Difference of Two Proportions

We can also use hypothesis testing to compare two proportions. This test is most common when you want to see if a population proportion is the same in two independent populations. For example, maybe we want to see if the proportion of men in the United States who have high trust in scientists (p_1) is different than the proportion of women in the United States who have high trust in scientists (p_2). Our two hypotheses are

$$H_0 : p_1 - p_2 = 0, H_1 : p_1 - p_2 \neq 0. \tag{6.7}$$

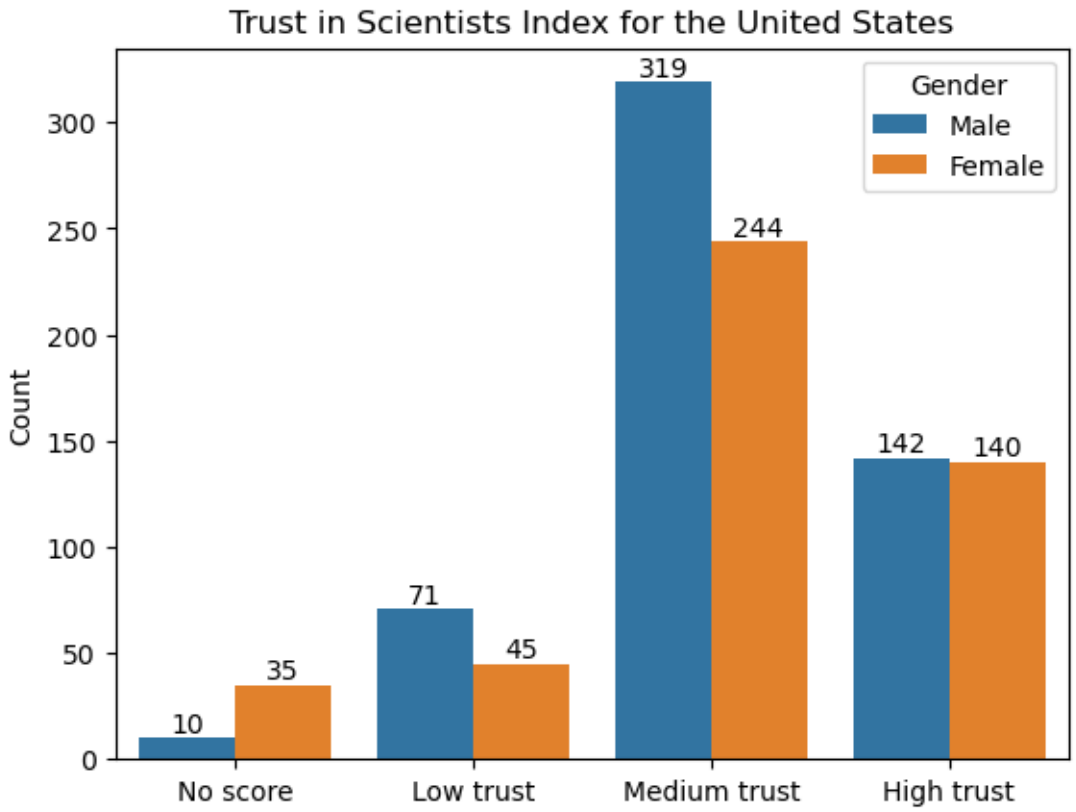
The main difference between this set of hypotheses and the previous set of hypotheses is that we aren't making a claim about whether the difference is bigger or smaller than a given value, just that it is different from that value. This will come into play later on when we quantify our evidence using the P-value.

Just as before, we will use the sample proportions of men and women from our survey to make a determination between the two hypotheses. Let's take a look at these:

```
# divide into two data sets
wgm_m = wgm_usa[wgm_usa['Gender'] == "Male"]
wgm_f = wgm_usa[wgm_usa['Gender'] == "Female"]

# number of successes (i.e., people with high trust in scientists)
count_m = len(wgm_m[wgm_m["Trust_Index"] == "High trust"])
count_f = len(wgm_f[wgm_f["Trust_Index"] == "High trust"])

# sample size (i.e., the total number of observations)
nobs_m = len(wgm_m["Trust_Index"])
nobs_f = len(wgm_f["Trust_Index"])

# compute sample proportion
p_hat_m =  count_m / nobs_m
p_hat_f = count_f / nobs_f
print(f"The sample proportion for men is {p_hat_m:.3f}. There are {nobs_m} total␣
 ↪observations for men.")
print(f"The sample proportion for women is {p_hat_f:.3f}. There are {nobs_f} total␣
 ↪observations for women.")
```

```
The sample proportion for men is 0.262. There are 542 total observations for men.
The sample proportion for women is 0.302. There are 464 total observations for␣
 ↪women.
```

```
# get dataframe
levels = wgm_usa[["Gender", "Trust_Index"]].value_counts().reset_index(name="Count")

# create bar plot
ax = sns.barplot(
    data = levels,
    x = "Trust_Index",
    y = "Count",
    hue = "Gender",
    order = ["No score", "Low trust", "Medium trust", "High trust"]
)
for container in ax.containers:
    ax.bar_label(container)
ax.set_xlabel("")
ax.set_title("Trust in Scientists Index for the United States");
```



Overall, in our sample we see a difference of about 0.04 between the proportions of men and women who have high trust in scientists. Again we ask, is that a large difference?

Recall that if our sample sizes n_1 and n_2 are large enough,

$$\hat{p}_1 - \hat{p}_2 \sim N\left(p_1 - p_2, \sqrt{\frac{p_1(1-p_1)}{n_1} + \frac{p_2(1-p_2)}{n_2}}\right). \tag{6.8}$$

and hence

$$Z = \frac{\hat{p}_1 - \hat{p}_2 - (p_1 - p_2)}{\sqrt{\frac{p_1(1-p_1)}{n_1} + \frac{p_2(1-p_2)}{n_2}}} \sim N(0,1). \tag{6.9}$$

The null hypothesis tells us what to fill in for $p_1 - p_2$, but it doesn't hypothesize directly about the values of p_1 or p_2. However, it does hypothesize that $p_1 = p_2$, and it that is true, then our estimates $\hat{p}_1$ and $\hat{p}_2$ are estimating the same thing. In that case, we can combine them to estimate the variance. Define p to be the common value of $p_1 = p_2 = p$. Then the variance of our test statistic is

$$\sqrt{\frac{p_1(1-p_1)}{n_1} + \frac{p_2(1-p_2)}{n_2}} = \sqrt{\frac{p(1-p)}{n_1} + \frac{p(1-p)}{n_2}} = \sqrt{p(1-p)\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}, \tag{6.10}$$

and we can estimate p using

$$p \approx \hat{p} = \frac{\text{num. of male respondents with high trust in scientists} + \text{num. of female respondents with high trust in scientists}}{\text{num. of male respondents} + \text{num. of female respondents}}. \tag{6.11}$$

We call $\hat{p}$ the **pooled sample proportion**.

```
p_hat = (count_m + count_f) / (nobs_m + nobs_f)
print(f"The pooled sample proportion is {p_hat:.3f}.")
```

```
The pooled sample proportion is 0.280.
```

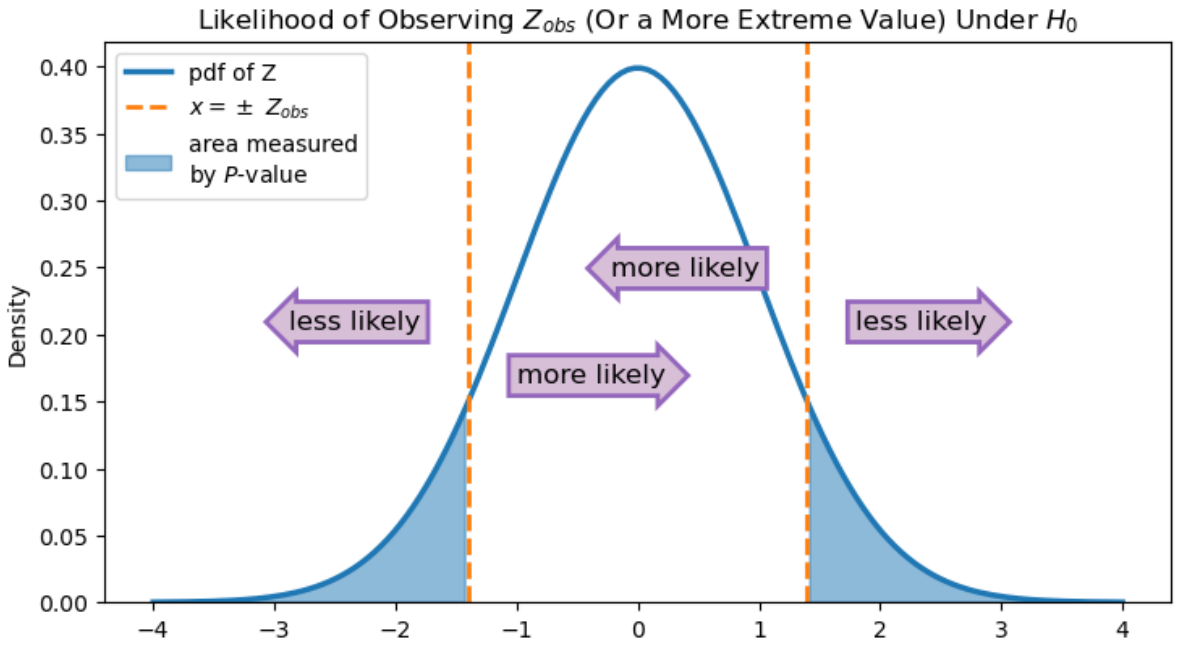
Now, using the formula for Z *(6.9)* above, we can calculate the test statistic.

```
z_obs = (p_hat_m - p_hat_f) / np.sqrt( p_hat*(1-p_hat)*(1/nobs_m + 1/nobs_f))
print(f"The observed value of the test statistic Z is {z_obs:.3f}.")
```

```
The observed value of the test statistic Z is -1.399.
```

Now that we have the test statistic, we can quantify the evidence that we have in relation to our two hypotheses. Since the alternative hypothesis is just interested in finding differences from 0, then values of Z that are a lot above 0 *or* a lot below 0 would be evidence against the null hypothesis.

In other words, if $p_1 - p_2$ really is 0, then Z will be close to 0 most of the time. If it's not, then it is more likely that Z is far from 0. We can quantify this through the P-value, which again is calculating the probability that we would observe a value a least as far from 0 as the one that we did. We double the value from the CDF here to get the P-value since "far away" includes both the positive and negative directions.



In other words, using the CDF will calculate the blue area under the curve in only one of those sections. So we double it to get the total area of the blue shaded region. Let's see what this P-value is:

```
2*(scipy.stats.norm(0,1).cdf(z_obs))
```

```
0.1619277283048255
```

This P-value is fairly large (> 0.05), indicating that if the null hypothesis is true, it's not unlikely that we would observe a difference of 4% in our sample. So we fail to reject the null hypothesis because we do not find evidence in our data of a difference of high trust in scientists between men and women in the United States.

6.3.3 Using Python for the Test

Of course, we're here to see how to perform these tests using Python code! The function to perform the test is the same as the one from the *previous section*, except this time the arguments `count` and `nobs` each take a list containing the relevant information from both of our samples. We have changed the value of the argument `alternative` to account for the fact that we are performing a `two-sided` test. Again, the output of the test is stored in `stat` and `pval`, and we have provided code to print them out in a nice way.

```
p0 = 0 # proportion under null hypothesis
(stat, pval) = sm.stats.proportion.proportions_ztest(
    count = [
        count_m, # number of successes for sample 1
        count_f, # number of successes for sample 2
    ],
    nobs = [
        nobs_m, # number of observations for sample 1
        nobs_f, # number of observations for sample 2
    ],
    value = p0,
    alternative = "two-sided",
    prop_var = False, # use pooled sample proportion to calculate variance for test␣
 ↪statistic
)
print(f"Z-test for difference of two proportions: test statistic is {stat:.3f}, P-
 ↪value is {pval:.3f}.")
```

```
Z-test for difference of two proportions: test statistic is -1.399, P-value is 0.
 ↪162.
```

These values match up with what we did before, so the conclusion is the same! Since we are failing to reject the null hypothesis, we think that the difference in means is actually 0, and there is no need to calculate a confidence interval. However, in other cases you might like to have access to it, so we will demonstrate that here. In the case of a difference of two proportions from independent samples, we use the `statsmodels` function `confint_proportions_2indep`. Notice that 0 is in the interval, confirming our conclusion.

```
conflevel = 0.95
(lower_ci, upper_ci) = sm.stats.proportion.confint_proportions_2indep(
    count1 = count_m, # number of successes for sample 1
    count2 = count_f, # number of successes for sample 2
    nobs1 = nobs_m, # number of observations for sample 1
    nobs2 = nobs_f, # number of observations for sample 2
    compare = 'diff', # compare differences of proportions
    alpha = 1-conflevel
)
print(f'{int(confidence_level*100)}% confidence interval is ({lower_ci:.3f}, {upper_
 ↪ci:.3f}).')
```

```
95% confidence interval is (-0.096, 0.016).
```

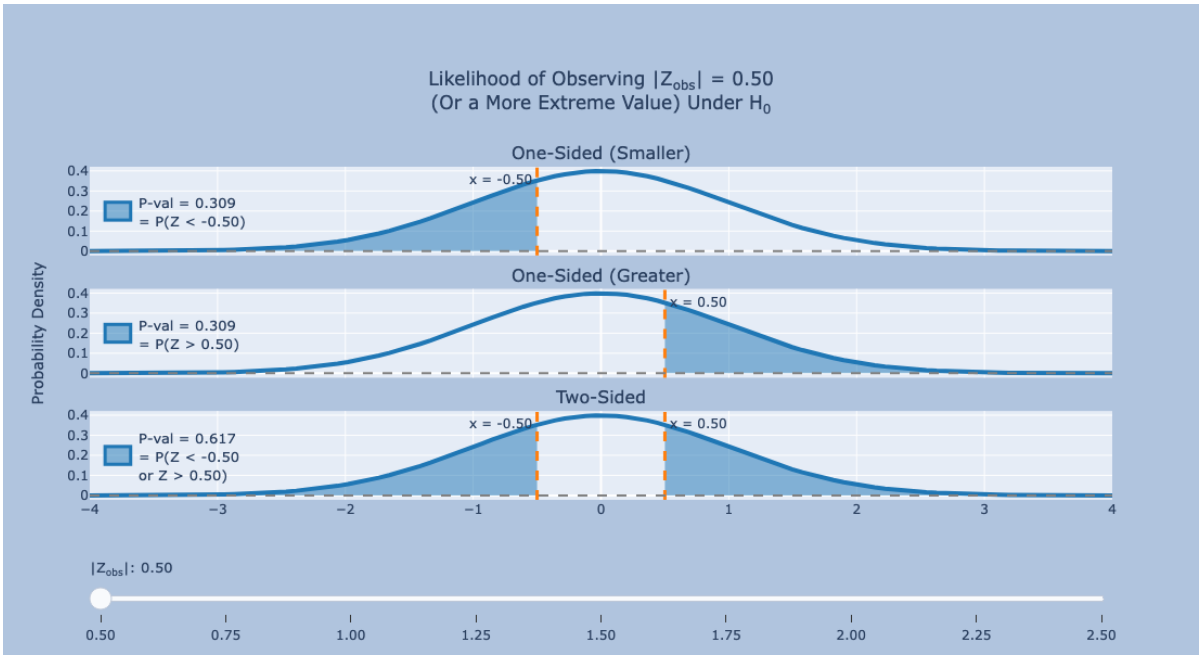
6.3.4 Further Exploration of P-Values

To improve your understanding of one-sided and two-sided P-values and how they are related to the observed value of your test statistic, you can play around with the interactive widget below.

Drag the slider to select different values of Z_{obs}. Compare the one-sided and two-sided P-values, and notice how the areas represented by the P-values shrink as Z_{obs} gets further and further away from 0, the mean of the standard normal distribution.

Note

Please visit the web version of this section to engage with this interactive resource.



6.3.5 Exercises

Exercise 1

To get some practice, we are going to look at trust in scientists in a different country.

1. Pick another country (perhaps your country of origin or another country you are interested in) and filter the observations to just include responses from that country. For example, here is how to do that for the United States.

```
wgm_usa = wgm[wgm['Country'] == "United States"]
```

2. Make a hypothesis about the proportion of all the people in your chosen country that have high trust in scientists. For example, do you think it is over 20%? Under 15%? Make any hypothesis you like.
3. Choose a level of α.
4. Compute the value of $\hat{p}$.
5. Run a hypothesis test using `statsmodels`.

6. Write your conclusion in a complete sentence. Be sure to report the test statistic and the P-value. If you found a significant result, give a confidence interval for the proportion.

Exercise 2

1. Pick a second country of interest.
2. Write hypotheses to test whether the proportion of people in the first country with high trust in scientists is higher than the same proportion in the second country.
3. Choose a level of α.
4. Run the hypothesis test using `statsmodels`.
5. Write your conclusion in a complete sentence. Be sure to report the test statistic and the P-value. If you found a significant difference, give a confidence interval for the difference in proportions.

```
# import libraries
# Always run this cell first!
import numpy as np
import pandas as pd

import scipy
import statsmodels.api # appear to need to import the api as well as the library␣
↪itself for the interpreter to find the modules
import statsmodels as sm

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import plotly.graph_objects as go
import plotly.offline
plotly.offline.init_notebook_mode(connected=True) # make plotly work with Jupyter␣
↪Notebook using CDN
```

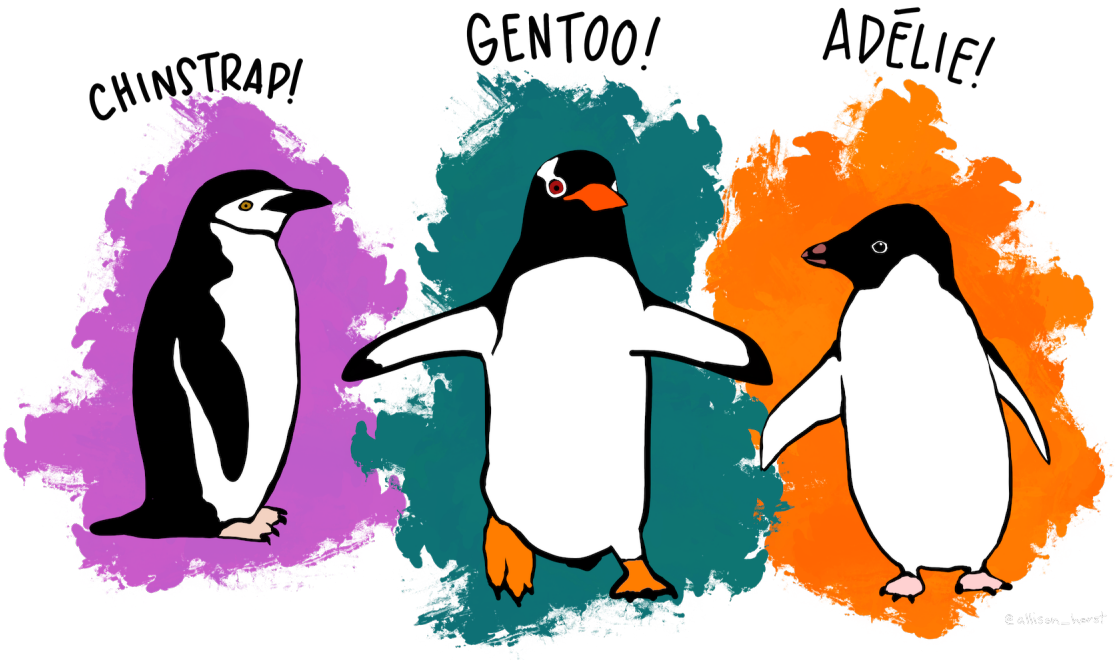
6.4 Hypothesis Testing for Numerical Data

In the *last section*, we learned about the statistical test of hypotheses, which tells us how we can choose between two statistical models. We did this in the context of categorical data, where we hypothesized about population proportions.

In this section, we apply the same theory to numerical data to hypothesize about population means and variances.

6.4.1 The Palmer Penguins

The Palmer Penguins dataset, created by Allison Horst, Alison Hill, and Kristen Gorman, contains physical measurements for 344 adult penguins observed near the Palmer Station in Antarctica. The penguins come from 3 different species: Chinstrap, Gentoo, and Adélie. The data were collected by Kristen Gorman and the Palmer Station LTER Program.



Artwork by @allison_horst, used with permission.

To load the data, you need to make sure that the `palmerpenguins` package is installed in your environment. We'll load the data as a Pandas data frame named `penguins`.

```
from palmerpenguins import load_penguins

penguins = load_penguins()
penguins.sample(5).head() # display 5 randomly sampled rows
```

```
       species     island  bill_length_mm  bill_depth_mm  flipper_length_mm  \
29      Adelie     Biscoe            40.5           18.9              180.0
300  Chinstrap      Dream            46.7           17.9              195.0
184     Gentoo     Biscoe            45.1           14.5              207.0
131     Adelie  Torgersen            43.1           19.2              197.0
148     Adelie      Dream            36.0           17.8              195.0

     body_mass_g     sex  year
29        3950.0    male  2007
300       3300.0  female  2007
184       5050.0  female  2007
131       3500.0    male  2009
148       3450.0  female  2009
```

Using this data, we are going to explore the question of whether the Gentoo penguins at the Palmer Station differ significantly in body mass from the average Gentoo penguin body mass found in the scientific literature. (This example was inspired by Erin Meyer-Gutbrod.)

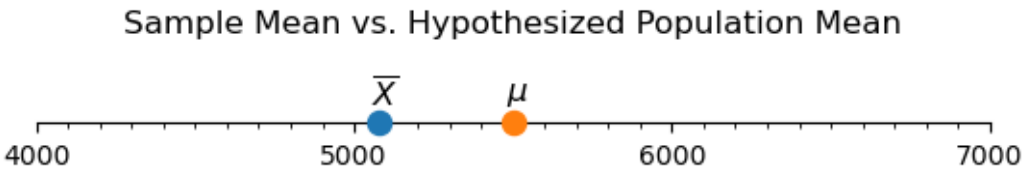
We can find more information about the worldwide average body mass of the Gentoo penguin by using the Encyclopedia of Life. Based on this page, it looks like the average body mass is 5500 grams. If we let μ be the average body mass of Gentoo penguins in the Palmer Station area, our hypotheses are

$$H_0 : \mu = 5500, H_1 : \mu \neq 5500. \tag{6.12}$$

What about the mean body mass for penguins in our sample data?

```
gentoo = penguins[penguins['species'] == 'Gentoo']
sample_mean = gentoo['body_mass_g'].mean()
print(f"The mean Gentoo body mass in the sample data is {sample_mean:.3f}.")
```

```
The mean Gentoo body mass in the sample data is 5076.016.
```



Though we could introduce a test based on the standard normal distribution—as we did in the *previous section*—in the case of numerical data, we are going to utilize the **T-test** instead. As the name suggests, this test is based on the Student's T-distribution. We do this because the T-test will apply both to the cases when the sample size is large and to the cases when it is relatively small.

6.4.2 The Student's T-Distribution

When hypothesizing about a population proportion, the null hypothesis gives us a value of p that determines both the mean and standard deviation of the null distribution. On the other hand, the CLT for sample means tells us that as long as n is large enough,

$$\overline{X} \sim N\left(\mu, \frac{\sigma}{\sqrt{n}}\right).$$

Unfortunately, if we make a hypothesis about μ, we are still left with estimating σ, the population standard deviation. If we have a good estimate for σ, perhaps from past experiments or expert knowledge, then this is no problem! If we don't know σ, our best estimate from the data will be to use S, the sample standard deviation.

When we standardize the test statistic, we then have

$$T = \frac{\overline{X} - \mu}{S/\sqrt{n}} \sim t_\nu \qquad \text{rather than} \qquad Z = \frac{\overline{X} - \mu}{\sigma/\sqrt{n}} \sim N(0,1). \tag{6.13}$$

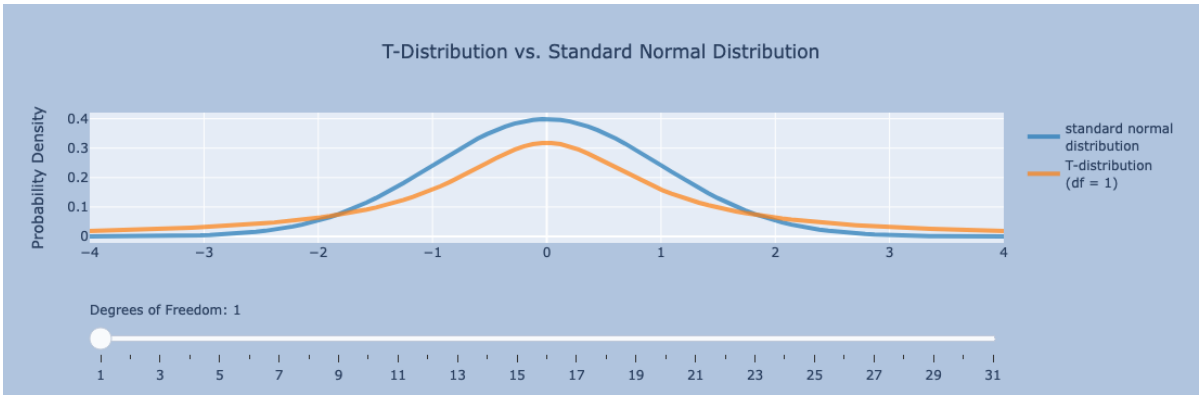
The good news is that T has a highly studied distribution called the Student's T-distribution. The T-distribution looks a lot like the standard normal distribution, but it has a wider spread, i.e., more probability in the "tails" of the distribution. This wider spread reflects the extra uncertainty of using the sample standard deviation S in place of the population standard deviation σ. The T-distribution has only one parameter, called "degrees of freedom", which is represented by the Greek letter ν (*nu*, pronounced like "new"). This is calculated by subtracting one from the sample size, i.e., $\nu = n - 1$.

We have glossed over an important detail here: The statistic $T = \frac{\overline{X}-\mu}{S/\sqrt{n}}$ has a T-distribution only if the data in the sample come from a normal distribution. Fortunately, if the sample size is large, we don't need to worry too much about this assumption thanks to the CLT. But if the sample size is small, this becomes especially important!

Play around with the widget below to get an idea of what the T-distribution looks like compared to the standard normal distribution. Drag the slider to select different degrees of freedom, and click the labels in the legend to hide or show different parts of the graph. Notice how the T-distribution resembles the standard normal distribution more closely as the degrees of freedom increase.

Note

Please visit the web version of this section to engage with this interactive resource.

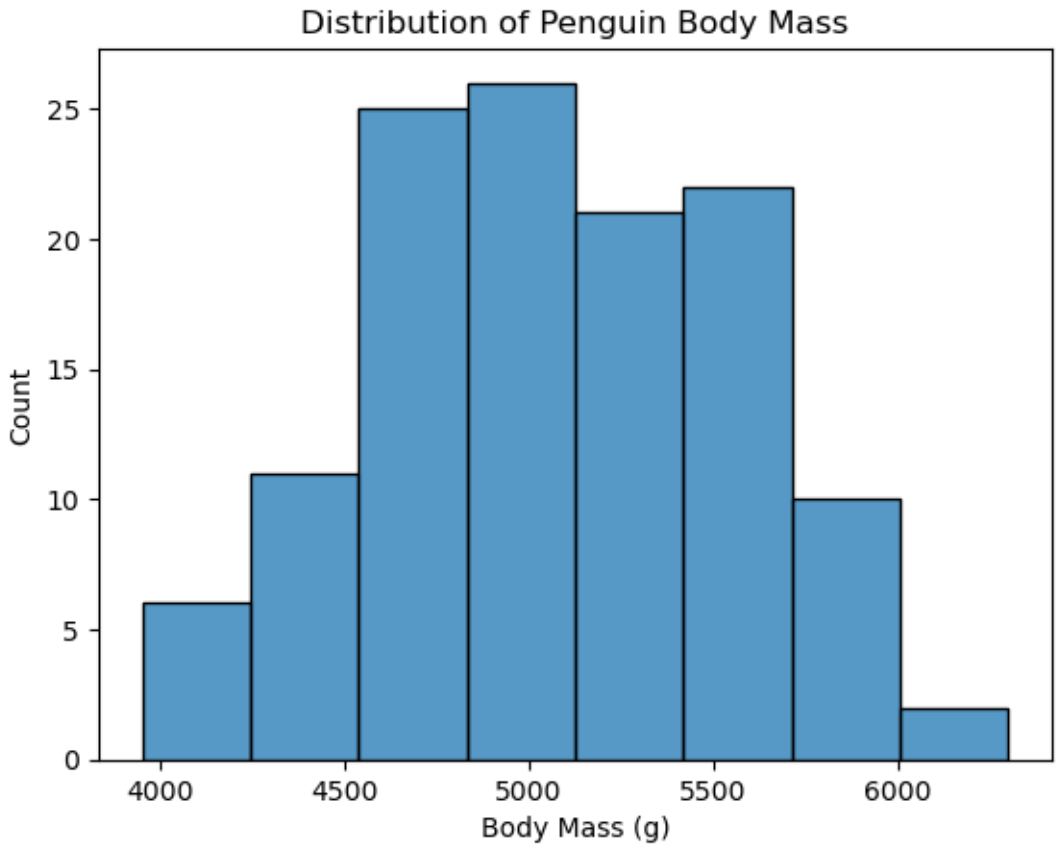


When n is large ($\geq$ 30 or so), then S should be close to σ, and the T-distribution more or less resembles the standard normal Z. So why use T at all? It turns out that the T-distribution can also be useful in cases when we do not have a large enough sample size to use the normal distribution! In this case, as we've already mentioned, we do need one extra assumption: that the population distribution has a normal distribution. This ensures that the sample average will also have a normal distribution. We'll come back to this with a small sample example later in the section.

6.4.3 The T-Test

Now back to our current example: to use a T-test, we need to check the assumption that the underlying data come from a normal distribution. We have $n = 119$ observations, which means we're working with a large sample case. Additionally, the histogram of the body mass of the Gentoo penguins is roughly mound-shaped, so using the T-test is appropriate for this situation.

```
# plot histogram
ax = sns.histplot(
    data = gentoo,
    x = 'body_mass_g',
)
ax.set_xlabel('Body Mass (g)')
ax.set_title('Distribution of Penguin Body Mass');
```



Now since we know the values of $\overline{X}$, S, and n, and the null hypothesis tells us what we assume μ to be, we can form our test statistic T based on the formula for T *(6.13)* above.

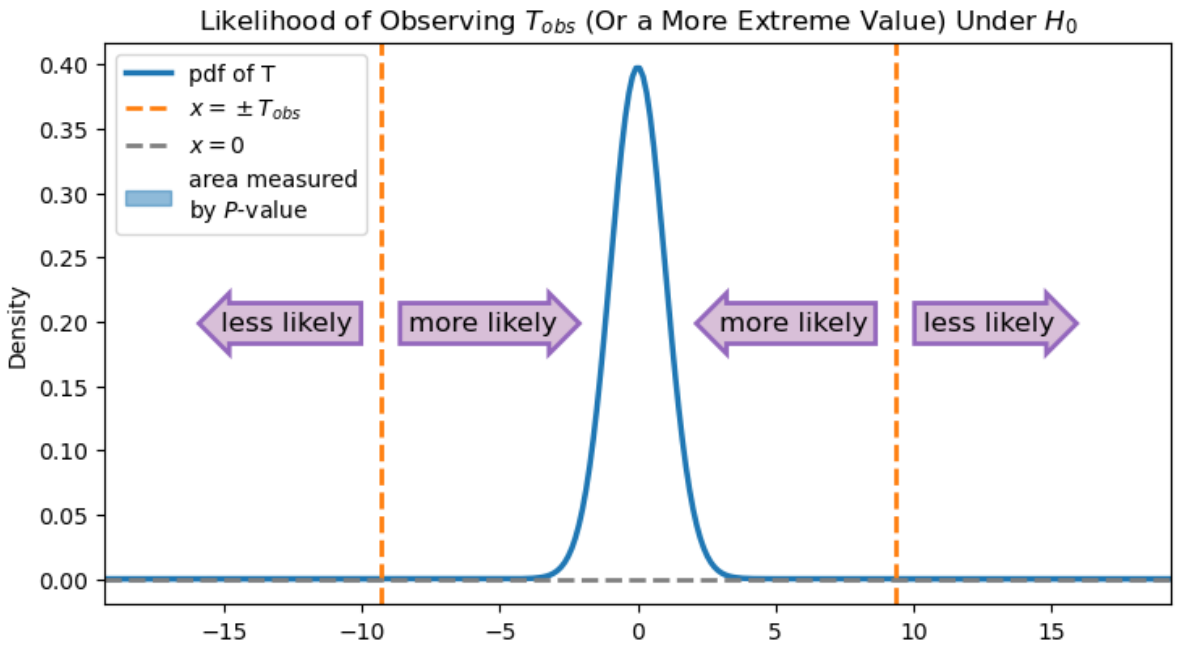
```
sample = gentoo['body_mass_g'].dropna() # drop missing values
sample_mean = sample.mean()
sample_sd = sample.std()
n = len(sample)
mu_0 = 5500 # hypothesized mean from H_0

T_obs = (sample_mean - mu_0) / (sample_sd / np.sqrt(n))

print(f"The observed value of the test statistic T is {T_obs:.3f}.")
```

```
The observed value of the test statistic T is -9.328.
```

If the population mean really is 5500 g, then we would expect $\overline{X}$ to be close to 5500 most of the time. Hence, we would expect T to be close to 0 most of the time.

On the other hand, if the null hypothesis is not true, then we would expect $\overline{X}$ to be different than 5500, and hence T will likely be much different from 0. Since we made a two-sided hypothesis, values of T that are much bigger than 0 *and* values of T much lower than 0 are both evidence against the null hypothesis.

We observed a very low value of T at -9.365. But again, we only get to observe one value of T and not the entire distribution! Is -9.365 different enough from 0 that we should start doubting that the null hypothesis is true? We can state this question as a probability: If the null hypothesis is really true, what is the chance that we would observe a value of T that is at least as far from 0 as the value we got (-9.365)?



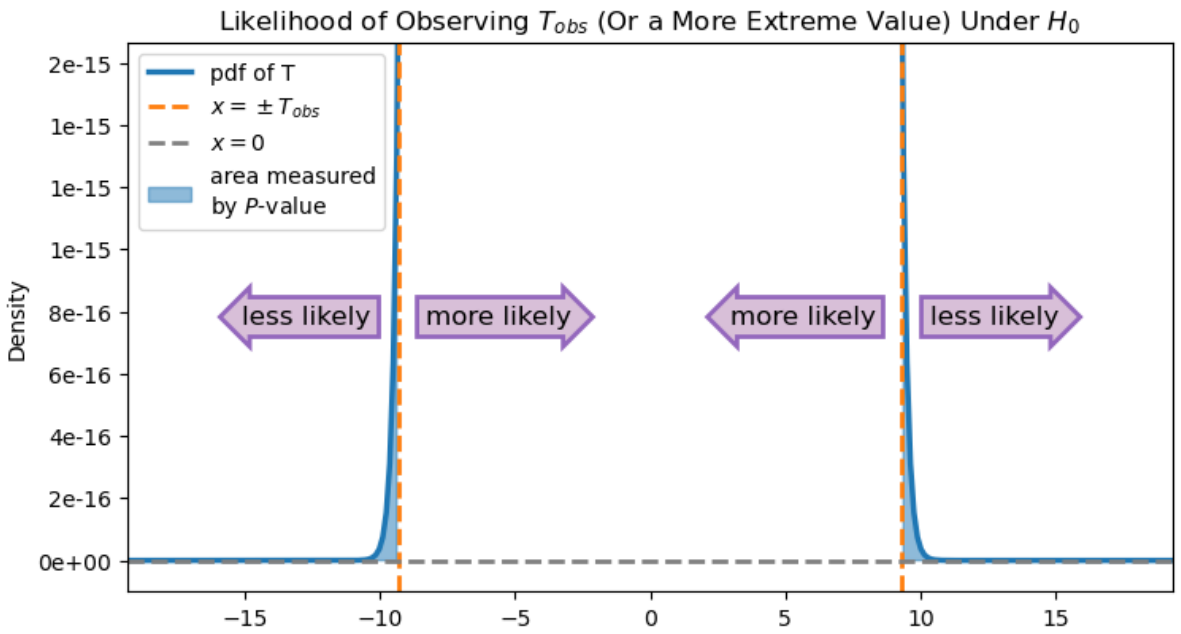

```
2 * scipy.stats.t.cdf(T_obs, df=n-1)
```

```
6.051182735596651e-16
```

This is the P-value again! Remember how we use the P-value to make a decision: If the P-value is greater than our specified threshold α, then our data is compatible with the null hypothesis, and we **fail to reject the null hypothesis**. If the P-value is less than our specified threshold α, then our test statistic is incompatible with the null hypothesis, and we will **reject the null hypothesis**.

In this case, the P-value is very small. As you can see from the image, the observed value is far from the hypothesized value, so the area under the curve in the "less likely" regions is minuscule. It's so small, in fact, that the area measured by the P-value in the figure above—the area between the density function and the line $x = 0$ that is to the left of $-T_{obs}$ or to the right of T_{obs}—isn't even visible at this scale.

Let's zoom in and take a closer look at this area:



As this graph shows, the area measured by the P-value is quite small (note the range of the y-axis).

Since the P-value is so small, we choose to reject the null hypothesis. There is evidence that the Gentoo penguins in this area are smaller than the average Gentoo penguin.

Why the big difference? Probably the Encyclopedia of Life body mass trait is specifically for adult Gentoo penguins, and our observations contain younger penguins. If we wanted to make a stronger, publishable conclusion we could use this as a jumping off point for further research.

If we were writing this test for a publication we could use language like this: "We conducted a one-sample t-test to compare the body mass of Gentoo penguins measured at the Palmer Station with the theoretical mean adult Gentoo body mass of 5500 g pulled from the Encyclopedia of Life trait bank. The penguin body mass measurements in this study were significantly smaller than the theoretical mean ($p = 6.05 \times 10^{-16}$)."

The T-Test in Python

Fortunately, Python has many of these statistical tests built in, so we don't have to calculate it all from scratch every time. In this case, we use the `scipy.stats` library, which includes the T-test as `ttest_1samp`. Here is code that performs the same test we just did. Verify that the answers match what we computed above!

```
result = scipy.stats.ttest_1samp(
    a = sample,
    popmean = 5500, # null hypothesis: mean is 0
    alternative = 'two-sided',
)
print(f"T-test for mean: test statistic is {result.statistic:.3f} with {n-1} degrees␣
↪of freedom.\nP-value is {result.pvalue:.3f}.")
```

```
T-test for mean: test statistic is -9.328 with 122 degrees of freedom.
P-value is 0.000.
```

The test again says that there is evidence that the population mean really is different from 5500. So what value will it take? To answer that, we can return to the confidence interval, this time calculated based on the T-distribution. How you do this might depend on what kind of software you are running.

Run the following code to check the version of `scipy` that you have installed.

```
print(scipy.__version__)
```

```
1.10.1
```

If your version of `scipy` is 1.10.0 or greater, you can compute the confidence interval by using the `confidence_interval` function associated with the result of the t-test we just ran:

```
# works for versions of scipy >= 1.10.0
confidence_level = 0.95
(lower_ci, upper_ci) = result.confidence_interval(confidence_level) # use the result␣
↪object from the t-test
print(f"95% confidence interval is ({lower_ci:.3f}, {upper_ci:.3f}).")
```

If your version of `scipy` is less than 1.10.0, you can compute the confidence interval by using the `t.interval` function instead. (It also works for versions greater than or equal to 1.10.0.) This function is similar to the `norm.interval` function we used *earlier*:

```
# works for versions of scipy < 1.10.0
confidence_level = 0.95
(lower_ci, upper_ci) = scipy.stats.t.interval(
    confidence_level,
    df = n - 1, # degrees of freedom of test statistic
    loc = sample_mean, # mean of test statistic
    scale = sample_sd/np.sqrt(n) # standard deviation of test statistic
)
print(f"95% confidence interval is ({lower_ci:.3f}, {upper_ci:.3f}).")
```

```
95% confidence interval is (4986.034, 5165.998).
```

We have 95% confidence that the average mass of Gentoo penguins in the Palmer Station area is between about 4986 and 5165 grams, much lower than the null hypothesis from before.

Exercise 1

Repeat the test above for the bill length of Gentoo penguins. This is found in the variable `bill_length_mm`.

1. Find the average bill length for Gentoo penguins from the Encyclopedia of Life. Use this information to form null and alternative hypotheses, making sure to note the units.
2. What is the sample size you have? Does the distribution of bill lengths appear to meet the normality assumption needed to perform the T-test?
3. Perform the test and report the test statistic and P-value.
4. What do you conclude? If you find a significant difference, give a confidence interval for average bill length.

6.4.4 Two-Sample T-Test

We can also use the T-test to examine the difference between two population means. Typically, this is done to find out if two populations have the same mean or not.

We can return to our Palmer Penguins example and look at whether the Adélie and Chinstrap penguins from this area have the same average body mass.

Letting μ_1 be the average body mass of Adélie penguins and μ_2 be the average body mass of Chinstrap penguins in the Palmer station area, our hypotheses are

$$H_0 : \mu_1 - \mu_2 = 0, H_1 : \mu_1 - \mu_2 \neq 0. \tag{6.14}$$

Let's look at our samples:

```
adelie = penguins[penguins['species'] == 'Adelie']
chinstrap = penguins[penguins['species'] == 'Chinstrap']

print(f"The mean Adélie body mass in the sample data is {adelie['body_mass_g'].
 ↪mean():.3f}.")
print(f"The mean Chinstrap body mass in the sample data is {chinstrap['body_mass_g'].
 ↪mean():.3f}.")
```

```
The mean Adélie body mass in the sample data is 3700.662.
The mean Chinstrap body mass in the sample data is 3733.088.
```

These are slightly different, but are they enough different for us to reject the null hypothesis? We need a test statistic to quantify this.

The Test Statistic

When the sample sizes n_1 and n_2 are large enough (typically both $n_1 \geq 30$ and $n_2 \geq 30$), we have

$$\overline{X}_1 - \overline{X}_2 \sim N\left(\mu_1 - \mu_2, \sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}\right), \tag{6.15}$$

and hence

$$\frac{\overline{X}_1 - \overline{X}_2 - (\mu_1 - \mu_2)}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \sim N(0, 1). \tag{6.16}$$

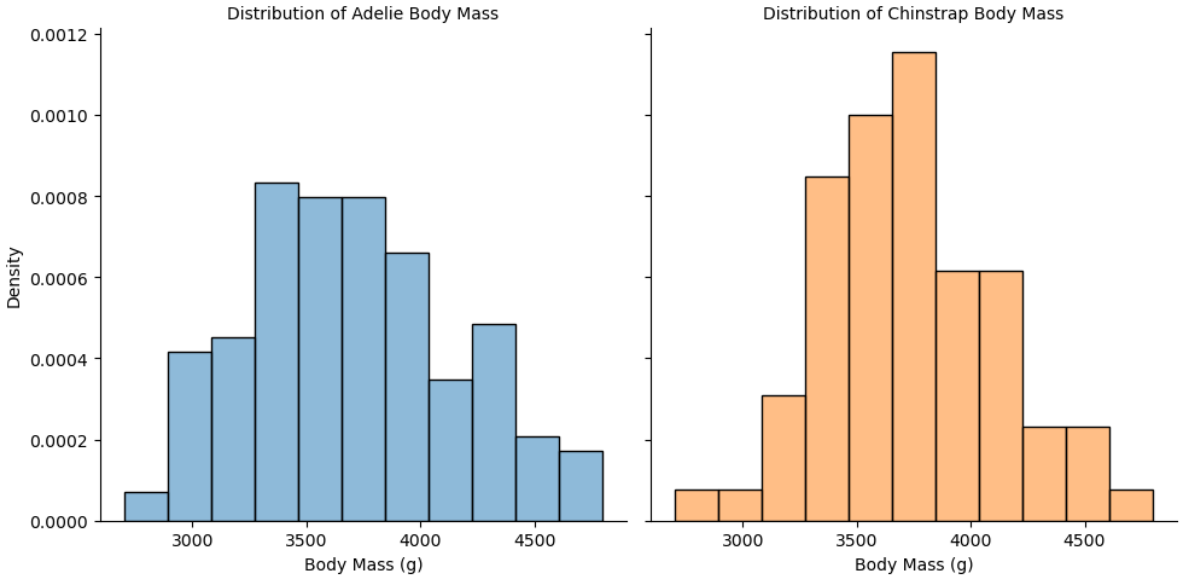
Just as before, the null hypothesis will tell us about $\mu_1 - \mu_2$, but not about σ_1 and σ_2, so we'll estimate them from the data using S_1 and S_2. Making that substitution for the standard deviations, we have a T-distribution:

$$\frac{\overline{X}_1 - \overline{X}_2 - (\mu_1 - \mu_2)}{\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}} \sim t_\nu, \tag{6.17}$$

where ν represents the degrees of freedom. The exact value of ν will be calculated by something called Welch's method, but we can let the software handle that details of the computation. You can keep in mind the intuition that if $S_1 \approx S_2$, then ν will be about $n_1 + n_2 - 2$. On the other hand, if S_1 and S_2 are very different, then $\nu \approx \min(n_1, n_2)$.

Now that we have two separate samples, the T-distribution requires that *both* come from normal distributions. There are sophisticated ways to check this, but we will just visually assess the histograms.

```
# plot histograms of adelie and chinstrap body mass, faceted by species
facetgrid = sns.displot(
    data = pd.concat([adelie, chinstrap]),
    x = 'body_mass_g',
    hue = 'species',
    col = 'species',
    kind = 'hist',
    stat = 'density',
    common_norm = False,
    legend = False,
)
facetgrid.set_titles('Distribution of {col_name} Body Mass')
facetgrid.set_xlabels('Body Mass (g)');
```



These look appropriately normal-ish, and the large sample gives us extra wiggle room in the assumptions of the T-test. Moreover, the distributions don't look too different, so we might start to expect that the true difference in means really is 0. So let's perform the test!

The function we will use is a function from `statsmodels` called `ttest_ind`. To use this function, we first need to create a `CompareMeans` object from our two samples. After we've done that, we will be able to call the `ttest_ind` function on this object to perform the actual T-test.

We set the argument `value` equal to the difference of means under our null hypothesis. We set the argument `alternative` to "two-sided" to indicate that we have an two-sided alternative hypothesis. (Other possible values are "larger" and "smaller".) Finally, we set the argument `usevar` to "unequal" to make sure that Welch's method is used to calculate the degrees of freedom.

```
# set up our two samples (removing all observations with missing values)
sample1 = adelie['body_mass_g'].dropna()
sample2 = chinstrap['body_mass_g'].dropna()

# create a CompareMeans object from the two samples
cm = sm.stats.weightstats.CompareMeans.from_data(sample1, sample2)

# perform the two-sample t-test
(stat, pval, df) = cm.ttest_ind(
    value = 0, # null hypothesis: difference in means is 0
```

(continues on next page)

(continued from previous page)

```
    alternative = 'two-sided', # two-sided alternative hypothesis
    usevar = 'unequal', # perform Welch's t-test
)

print(f'Results of T-test: test statistic is {stat:.3f} with {df:.3f} degrees of␣
 ↪freedom.\nP-value is {pval:.3f}.')
```

```
Results of T-test: test statistic is -0.543 with 152.455 degrees of freedom.
P-value is 0.588.
```

We do not find any evidence of a difference in average body mass between these two species, so we fail to reject the null hypothesis.

In this situation, we've failed to reject the null hypothesis, so we are not necessarily interested in computing a confidence interval for this difference (since we already suspect it to be 0). However, for future reference, here is how to compute a confidence interval associated with this test.

```
conflevel = 0.95

# compute confidence interval, using the same CompareMeans object as before
(lower, upper) = cm.tconfint_diff(
    alpha = 1-conflevel,
    alternative = 'two-sided',
    usevar = 'unequal', # perform Welch's t-test
)
print(f'{int(conflevel*100)}% confidence interval is ({lower:.3f}, {upper:.3f}).')
```

```
95% confidence interval is (-150.385, 85.533).
```

Note that this confidence interval contains 0. Since we have 95% confidence that the difference in the average mass of Adélie and Chinstrap penguins in the Palmer Station area is in this interval, this confirms our conclusion that we have no evidence to reject the null hypothesis.

Large Sample vs. Small Sample

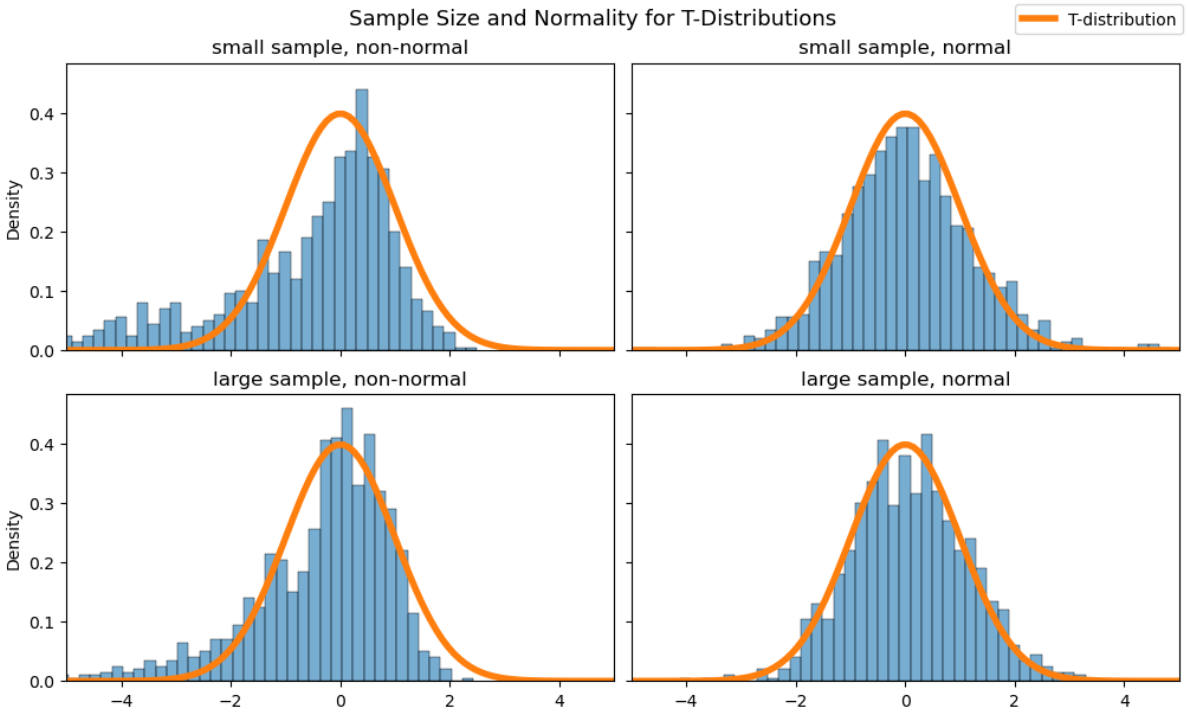
Why do we need the extra assumption of a normally distributed population in the case of a small sample? In non-technical terms, because we have less information in the data, we need more information from elsewhere (in this case, assumptions) to be able to make conclusions.

In the large sample case, the Central Limit Theorem tells us that $\overline{X}$ is normal. In the small sample case, if the sample comes from a normally distributed population, then $\overline{X}$ will again be normal.

Below you can find plots of the sampling distribution of sample averages from both normal and non-normal distributions, each with a population mean of 1 and a standard deviation of 2. The top row looks at small samples (size $n = 10$), one from a non-normal distribution (left) and the other from a normal distribution (right). We take the average and "studentize" according to the formula for *T (6.13)* above.

Look first at the upper left plot. The histogram of small-sample averages from the non-normal distribution do not fit well with a T-distribution on 9 degrees of freedom. However, the small-sample averages from the normal distribution in the upper right plot do!

In the bottom row you can find histograms of simulated large sample averages ($n = 50$), overlaid with the density function for a T-distribution on 49 degrees of freedom. The sample averages from both the non-normal (lower left) and normal (lower right) distributions end up fitting well with the T-distribution.



This shows us that, in order to use the T-test with a small sample, it's necessary to assume that our data comes from a normal distribution. But how we can tell if this assumption is true? This is a nuanced question. Because you don't have much data to start with, it's difficult to draw strong conclusions about the distribution it came from! Some textbooks will simply tell you to look for rules of thumb: as long as there are no outliers, or the graph looks roughly "mound-shaped", you're ok to proceed with the test. There are even hypothesis tests to quantify this or more sophisticated visuals to use, but they are outside of the scope of this chapter.

Robustness to Assumptions

Statisticians have determined that the T-test is fairly "robust" to the normality assumptions. In simple terms, robustness means that you can still trust the results of the test, even if your assumptions are only approximately met. In this case, if your data comes from something that is close to normal, the T-test will still work properly. If that feels too imprecise, that's ok! It can take some time to develop your understanding and intuition for this.

```
# import libraries
# Always run this cell first!
import numpy as np
import pandas as pd
import math

import scipy
import statsmodels.api # appear to need to import the api as well as the library␣
↪itself for the interpreter to find the modules
import statsmodels as sm

import matplotlib.pyplot as plt
from matplotlib import ticker
import seaborn as sns
%matplotlib inline
import plotly.graph_objects as go
import plotly.offline
```

(continues on next page)

(continued from previous page)

```
plotly.offline.init_notebook_mode(connected=True) # make plotly work with Jupyter␣
↪Notebook using CDN

# an extra function for plotting a straight line
def plot_abline(slope, intercept, color = None, x_name = "x", y_name = "y"):
    """Plot a line from slope and intercept"""
    axes = plt.gca()
    x_vals = np.array(axes.get_xlim())
    y_vals = intercept + slope * x_vals
    plt.plot(
        x_vals, y_vals, '--', color = color,
        label = f"${y_name} = {slope:.2f}{x_name} " + ("-" if intercept < 0 else "+")␣
↪+ f"{abs(intercept):.2f}$"
    )
```

6.5 Linear Regression

We next turn to explore another type of statistical decision-making tool, linear regression. Linear regression is a tool to allow us to explore the connections between 2 or more variables.

6.5.1 Simple Linear Regression

To begin, we'll look at simple linear regression, which models the linear association between two variables. This allows us to see how changes in one variable affect the other. To motivate the theory, we'll take a look at how statistics related to primary school education in a given country are associated with the literacy rate in that country.

World Bank Development Indicators

The World Development Indicators dataset, maintained by the World Bank, contains information about a wide variety of topics related to global development. The most recent version of the dataset, as of writing, provides over 1500 indicators for 217 economies from 1960 to 2021. The dataset is licensed under a Creative Commons Attribution 4.0 International License.

For our example, we will look at three statistics related to literacy. We will limit our analysis to the 89 countries that have valid data for all three statistics for at least one year in the time range 2017-2021. For each country, we will consider the most recent year from that time range for which all three statistics are available.

Let's take a look at this data, which is in a file called `world_bank_literacy.csv` (click here to download a copy).

```
wbl = pd.read_csv('world_bank_literacy.csv', index_col = 'country_code')
wbl.head()
```

```
                          country  year  adult_literacy_rate_(pct)  \
country_code
ALB                       Albania  2021                  98.449997
ARE          United Arab Emirates  2019                  95.526482
ARG                     Argentina  2018                  99.003868
ARM                       Armenia  2020                  99.788612
AZE                    Azerbaijan  2019                  99.795982
```

(continues on next page)

(continued from previous page)

```
              primary_school_completion_rate_(pct)  \
country_code
ALB                                       98.155830
ARE                                      111.938820
ARG                                       98.706253
ARM                                       93.165291
AZE                                      100.383339

              primary_school_enrollment_rate_(pct)
country_code
ALB                                       95.921783
ARE                                      114.412361
ARG                                      109.659172
ARM                                       91.235443
AZE                                       97.935349
```

Here is a brief summary of the information in the last three columns.

- The column labeled `adult_literacy_rate_(pct)` (World Bank indicator SE.ADT.LITR.ZS) contains the percentage of people age 15 and up in each country who are literate. The World Bank defines a person as literate if they can "read and write with understanding a short simple statement about their everyday life."
- The column labeled `primary_school_enrollment_rate_(pct)` (World Bank indicator SE.PRM.ENRR) contains the gross percentage of people enrolled in primary school in each country. This percentage is calculated by taking the total number of primary school students in a given country (no matter what their age is) and dividing it by the total number of people in the *official* age group for primary school in that country. Because some of the students in primary school might be older or younger than the official primary school ages, this percentage can be greater than 100%.
- The column labeled `primary_school_completion_rate_(pct)` (World Bank indicator SE.PRM.CMPT.ZS) contains the gross percentage of people enrolled in the *last* grade of primary school in each country. This percentage is calculated by taking the total number of non-repeating students in the last grade of primary school in a given country (no matter what their age is) and dividing it by the total number of people in the *official* age group for the last grade of primary school in that country. This percentage is not the actual rate of students who complete primary school, but it serves as a proxy (a variable we can measure that helps us approximate something else we can't measure) for that rate. As with `primary_school_enrollment_(pct)`, this percentage can be greater than 100% because some of the students in the last grade of primary school might be older or younger than the official age for that grade.

Let's suppose we are interested in literacy rates worldwide. We could begin our exploration by taking a look at some summary statistics:

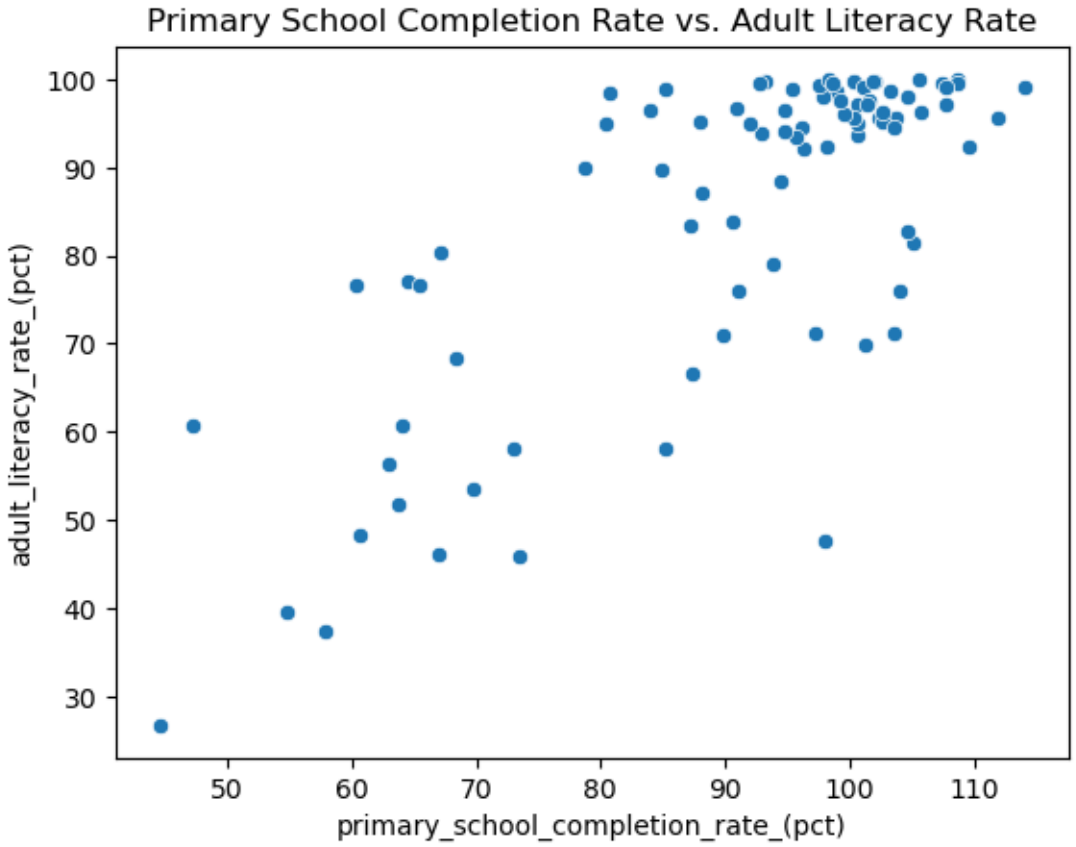
```
wbl['adult_literacy_rate_(pct)'].describe()
```

```
count    89.000000
mean     85.170694
std      18.214867
min      26.760000
25%      76.570518
50%      94.971130
75%      98.419998
max      99.999947
Name: adult_literacy_rate_(pct), dtype: float64
```

Note that since we take the average over countries *in the sample, the average literacy rate here also refers to an average country. Since the average rate doesn't take into account the population of any country, it is not reflective of the overall*

worldwide literacy rate.

An average country in our sample has a literacy rate of about 85%, and there is a fair bit of variation in the literacy rates. But what are some of the things that are connected with literacy rates in different countries? For example, maybe literacy rate is connected to the percentage of students in the country that complete primary schooling. We could explore this connection with a scatter plot:



It looks like there is some connection here! As primary school completion rate increases, so does the literacy rate. We might call this a **positive trend**. Plus, it looks like you might be able to draw a line through the middle of this cloud of points to describe the way the cluster of points moves. So we might also call this a **linear trend**. We can describe this relationship in more depth using the tools of linear regression.

Note that just because these two variables seem to have some pattern when plotted together, that doesn't mean that primary school completion rate *causes* the change in literacy rate. It just means that primary school completion rate is *associated* with the change in literacy rate. (We'll use the word **association** to describe any pattern or connection between two variables, because "correlation is not causation".)

Simple Linear Regression Models

Simple linear regression is the method we have of modeling the linear association between two variables. You may have learned in algebra that one common way to write the equation of a line is the *slope-intercept form* $b+mx$. In this equation, m is the slope, and the addition of b means that the line passes through the point $(0, b)$, which is called the intercept of the line.

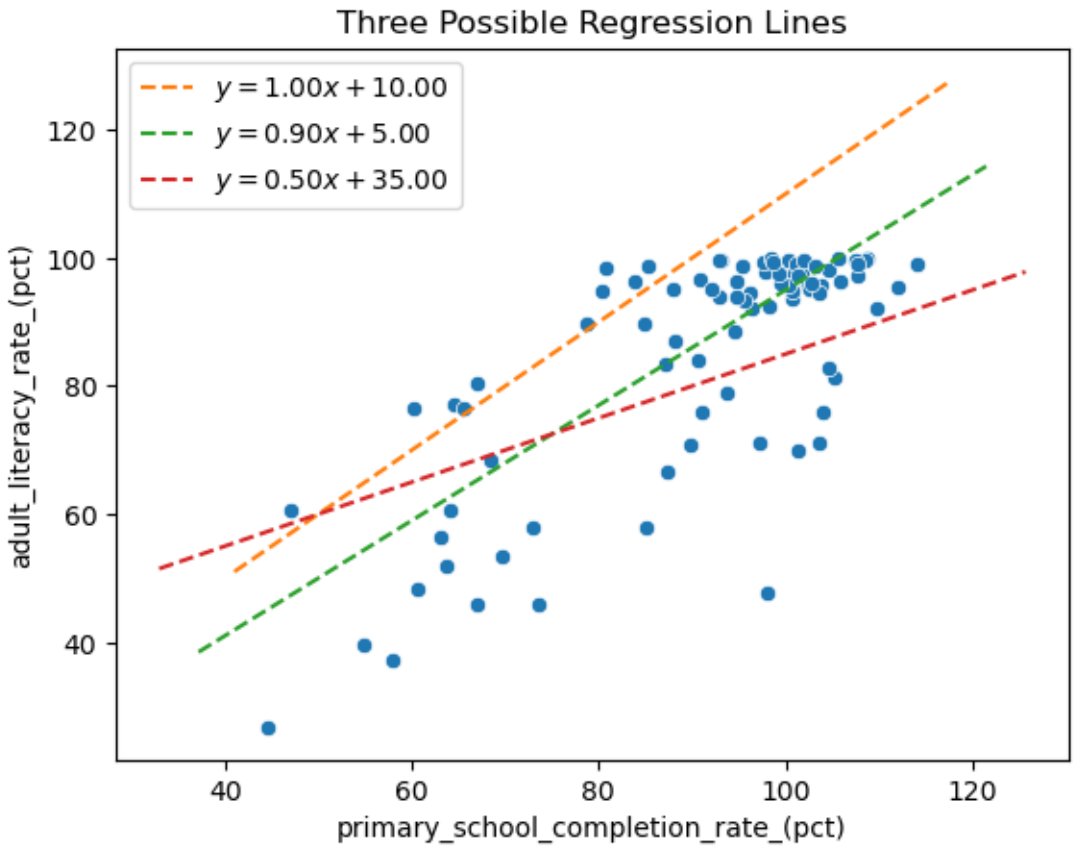
However, we can see from the *scatter plot* that our data points don't exactly fit a straight line. Instead, they seem to have some random deviations around a general trend line. We can write this *line-plus-random-error* idea using the mathematical equation

$$Y = \alpha + \beta x + \varepsilon. \tag{6.18}$$

This is another type of statistical model, called a linear regression model. It says that the value of (the random variable) Y is just the value along the line $\alpha + \beta x$ plus a random error ε. If we knew the slope β and intercept α of the line, then we could plug in the value of x to get a good guess for Y, though it would be off by some error.

You probably have many questions at this point, especially if this idea is new to you! But we want to first consider a big one: how do we figure out the equation for the line? There are multiple lines we could draw. Below are three of them. If you're following along in an interactive notebook, try playing around with the slope and intercept of the first line to see if you can make it fit the data as best you can.

```
# Make your own line by changing around these numbers:
slope = 1.0
intercept = 10.0

# add scatter plot of data
palette = iter(sns.color_palette())
sns.scatterplot(
    data = wbl,
    x = 'primary_school_completion_rate_(pct)',
    y = 'adult_literacy_rate_(pct)',
    color = next(palette)
)

# plot the lines with given slope and intercept
# use function defined in first cell
plot_abline(slope, intercept, color = next(palette))
plot_abline(.9, 5.0, color = next(palette))
plot_abline(.5, 35, color = next(palette))
plt.title("Three Possible Regression Lines")
plt.legend();
```



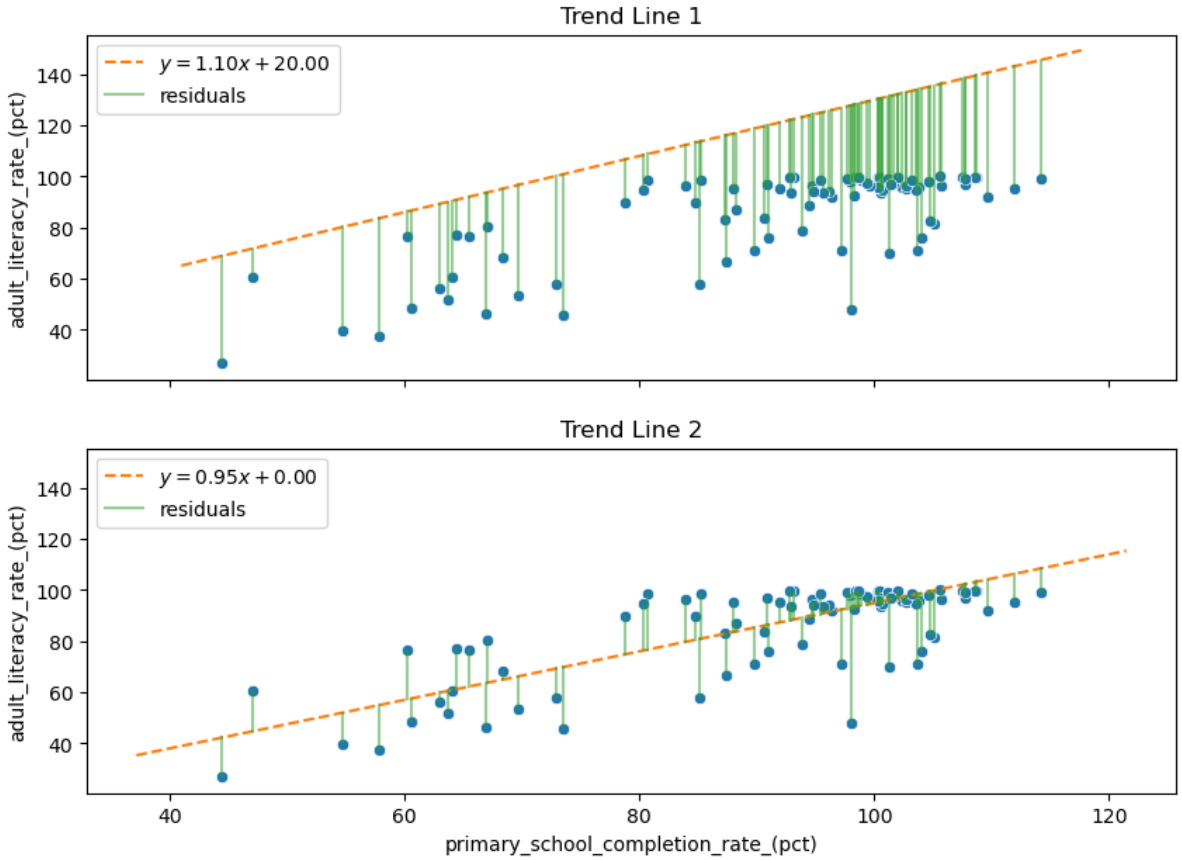
Which of these lines is best? And what would make a line the "best" one in this situation? The method of least squares, described below, is a way to answer both of these questions.

Minimizing the Sum of Squares Error

If you spent some time thinking about where the "best" line should go, you probably tried to get it somewhere in the middle of the cloud of points, so that it wasn't too far away from any one point. In that way it would follow the general trend, even if it didn't exactly match any of the data points.

We can make this idea more mathematical by considering something called the **residual**. For each country in our data set, we have a pair of primary school completion rates and literacy rates, which we could enumerate as pairs (x_i, y_i). Given any line $\hat{y} = a + bx$, the residual is the difference between the actual value y_i associated with x_i in the data, and the value $\hat{y}_i$ that lies along the line $a + bx_i$ (that is, the residual is $y_i - \hat{y}_i$). In other words, the residual measures how far our line is from passing through a certain data point.

Below we look at plots of two different lines and examine the residuals associated with them. The residual for each point is the vertical distance from the point to the line. Which line do you think better fits the trend of the points? And what is the difference between the residuals of the two trend lines?



The better trend line, Trend Line 2, has overall smaller residuals, so we might think that it would help to pick the line that minimizes the sum of the residuals. This is *almost* the right idea! However, the worse line, Trend Line 1, has only negative residuals. So in fact the sum of its residuals is a negative number that will be smaller (at least on the number line) than the sum of the residuals of the better line.

It turns out we don't care about the sign of the residual, only its magnitude. To solve this issue, we square each of the residuals and take the sum to form a **sum of squared errors (SSE)**:

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{6.19}$$

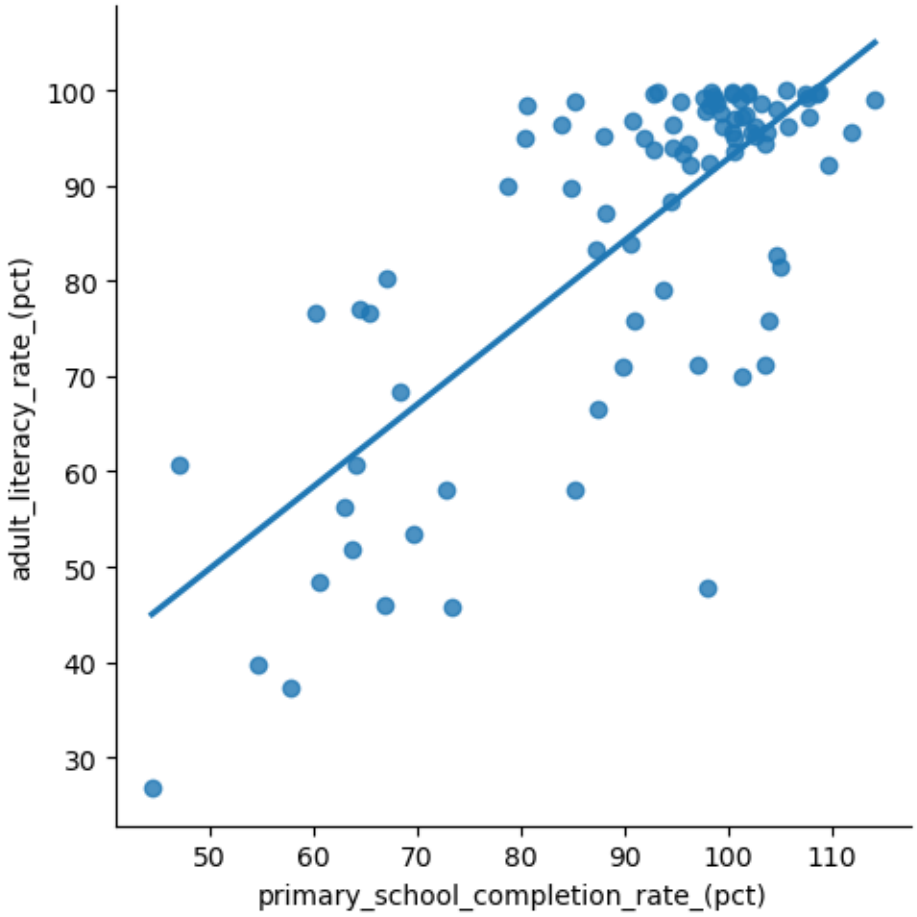
In the end, we want to find the slope and intercept of the line that minimizes this quantity. Since we are minimizing the sum of squared errors, this best-fit line also goes by the name **least-squares regression** line. Solving this problem looks like it could be tricky, but it turns out that the solution is straightforward when we employ the help of a little bit of calculus. Or, even nicer (in the author's opinion) is to use *a little algebra*. We do need to assume that the random error ε is normally distributed, but we will gloss over this assumption for now. You can read more about how to find the line of best fit elsewhere if you're interested, but here we'll simply find the solution using code.

Why take the sum of the squares of the residuals instead of the sum of the absolute values of the residuals? One reason is mathematical: there is always a unique global minimizer to the SSE. Since squaring a large number will make it larger, this method penalizes very large residuals. Another reason is statistical: it turns out the slope and intercept of the least squares solution have special mathematical properties which make them the *maximum likelihood estimators* of the model's true slope and intercept. But we won't look into that here!

Finding the Least Squares Line

First, we can easily use software to get a visual of the least squares regression line using `seaborn`'s `lmplot`:

```
sns.lmplot(
    data = wbl,
    x = 'primary_school_completion_rate_(pct)',
    y = 'adult_literacy_rate_(pct)',
    ci = None
);
```



But what if we want to be more precise and want the equation of the line? For this, we'll use the `OLS` function from `statsmodels`.

The `OLS` function is found in `statsmodels.regression.linear_model`, and it takes two arguments: the dependent variable x and the independent variable y. Since we want a line that has an intercept, we plug our x variable into the `add_constant` function to make sure we get the intercept term. This creates our "model object", which we named `lin_mod`.

The linear model object `lin_mod` has a `fit()` function, which performs the least squares algorithm to find the equation of the line of best fit. The results of the algorithm can be printed out in a summary, which you see below. This is going to display a lot of output, but we'll walk through it slowly!

```
x = 'primary_school_completion_rate_(pct)'
y = 'adult_literacy_rate_(pct)'
```

(continues on next page)

(continued from previous page)

```
lin_mod = sm.regression.linear_model.OLS(
    wbl[y], # the dependent variable (y)
    sm.tools.tools.add_constant(wbl[x]) # the independent variable (x)
)

results = lin_mod.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     adult_literacy_rate_(pct)   R-squared:                       0.
 ↪567
Model:                                   OLS   Adj. R-squared:                  0.
 ↪562
Method:                        Least Squares   F-statistic:                     ␣
 ↪114.0
Date:                       Thu, 20 Jul 2023   Prob (F-statistic):           1.70e-
 ↪17
Time:                               16:47:33   Log-Likelihood:                -346.
 ↪82
No. Observations:                         89   AIC:                             ␣
 ↪697.6
Df Residuals:                             87   BIC:                             ␣
 ↪702.6
Df Model:                                  1
Covariance Type:                   nonrobust
================================================================================================
                                         coef    std err          t      P>|t|  ␣
 ↪   [0.025      0.975]
------------------------------------------------------------------------------
 ↪---------------------
const                                  6.6714      7.462      0.894      0.374  ␣
 ↪   -8.160      21.503
primary_school_completion_rate_(pct)     0.8615      0.081     10.677      0.000  ␣
 ↪    0.701       1.022
==============================================================================
Omnibus:                       12.535   Durbin-Watson:                   1.772
Prob(Omnibus):                  0.002   Jarque-Bera (JB):               13.251
Skew:                          -0.842   Prob(JB):                      0.00133
Kurtosis:                       3.861   Cond. No.                         540.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly␣
 ↪specified.
```

There is a lot of information here, most of which we'll explain later on. For now, let's see how to get the information about the slope and the intercept of the line. You can find these values in the second table under the column heading `coef`. From this, we can see that the intercept (`const`) is 6.6714, and the slope associated with the primary school completion rate is 0.8615. We can write that in slope-intercept form as the line

$$\hat{y} = 6.6714 + 0.8615x \tag{6.20}$$

Now that we have this line, how can we understand how to use it? First, the slope tells us how the changes in x are connected to changes in $\hat{y}$. Here, an increase of 1% in the primary school completion rate (x) is associated with an average increase of about .865% in the literacy rate (y).

The intercept is a bit harder to interpret in this context, but it's saying that in this model, a country with 0% primary school completion rate will, on average, have a literacy rate of 6.6714%. Of course, a 0% completion rate is much outside the range of values we have for x, and we don't recommend using the model to make predictions for that range of literacy rates! (That's called *extrapolation*.)

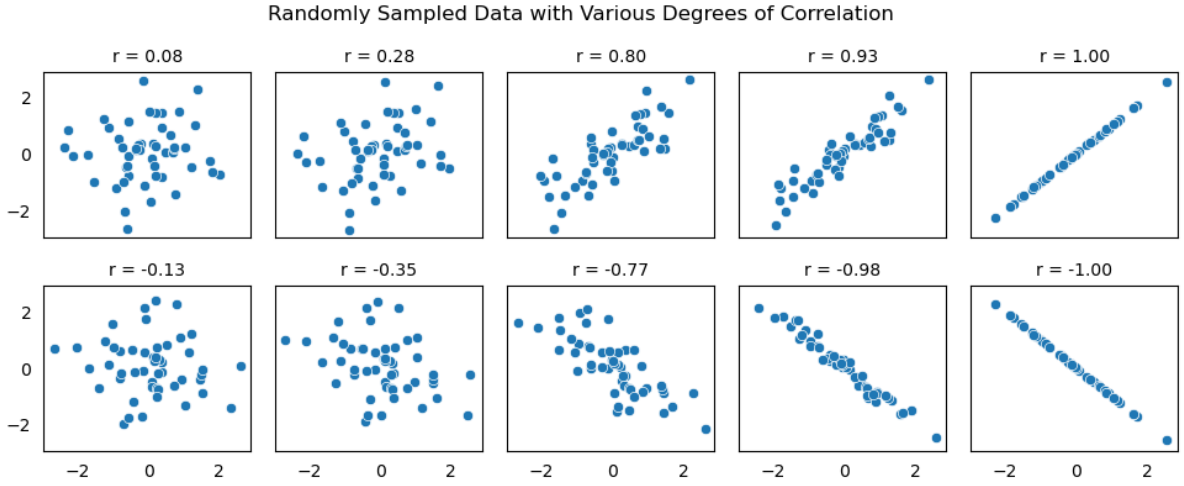
Next, we want to look at how well this model fits the data.

R^2 and Goodness of Fit Tests

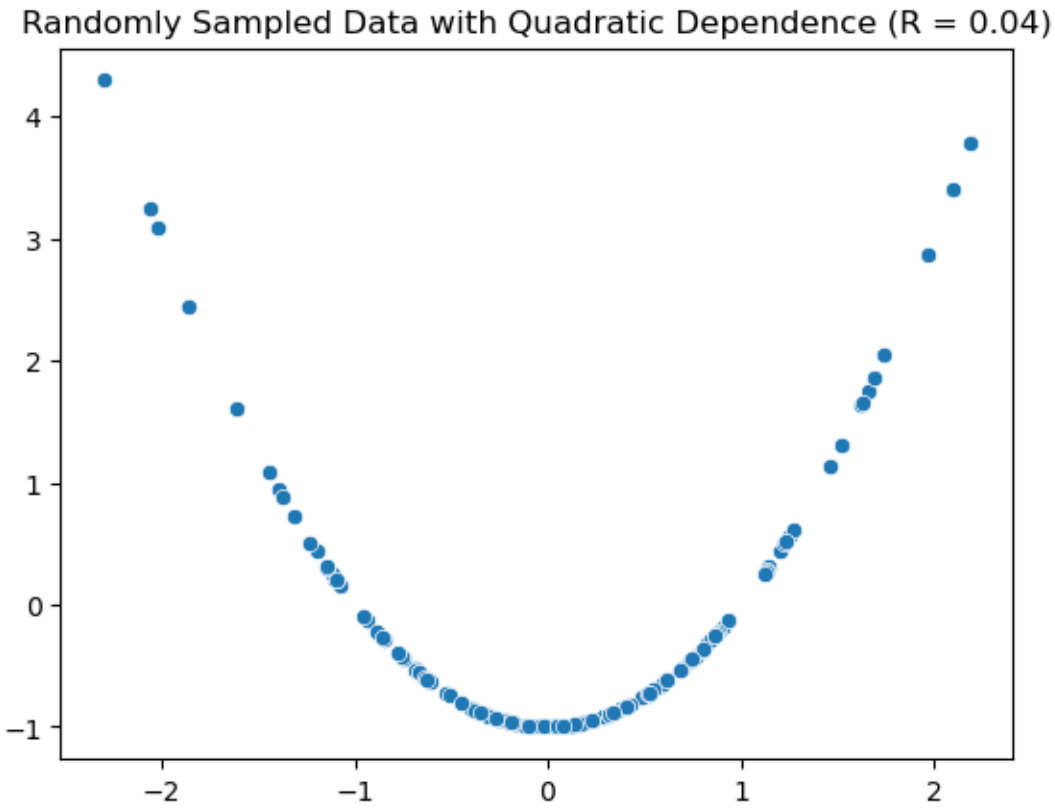
We mentioned before that we are using linear regression to *model* the relationship between two variables. But this only works if these variables actually have a linear relationship! Maybe they are actually related in a nonlinear way, or not related at all.

We can get some idea of this by looking at the *scatter plot*, but we can also quantify the strength of a linear relationship through a number called the **correlation coefficient** r. The number r is a value between -1 and 1 that measures the strength of the linear association between two variables.

What's a large value of r versus a small one? It depends on the problem! And as you get more comfortable with this, you'll get a sense for it. Below we'll take a closer look at how to understand this. As you look at these different plots, which would you say have a strong linear association?



When we emphasize that r measures *linear* association, this means that two variables might have a very small value of r, but still have some *nonlinear* association. Notice in the following scatter plot that the points have a quadratic relationship, but the measure of correlation is close to 0.



There are a few ways to calculate r in Python. First, we could calculate it directly using the `pearsonr` command from `scipy.stats`:

```
x = wbl['primary_school_completion_rate_(pct)']
y = wbl['adult_literacy_rate_(pct)']
r = scipy.stats.pearsonr(x, y)[0]

r
```

```
0.7531156213152443
```

Second, we also get the magnitude of r in the output of our linear model. If you look above, you see that the output of the linear model says `R-squared:  0.567`. This actually provides you with a value r^2, which is simply the square of the correlation coefficient r that we calculated above:

```
r**2
```

```
0.5671831390690465
```

Even though it doesn't tell us if the trend is positive or negative, r^2 is often preferred to r because it has a simple interpretation. The value of r^2 quantifies the **percentage of variation in Y that is explained by the linear model.**

In the context of our problem, using a linear model with the primary school completion rate explains 56.7% of the variation in literacy rates between the countries. More simply, the primary school completion rate explains some of the differences in literacy rates between countries, but not all, and that amount is quantified as 56.7%.

Is this a small or large amount? One way to quantify how well the model fits is through a *Goodness of Fit Test*. Without giving all of the details, this is a hypothesis test that determines if the value of r is significantly different from 0. In terms of hypotheses, the null hypothesis is that the model is not a good fit, versus the alternative that it is a good fit.

The test statistic has a special type of distribution called an F distribution, and its value can be found on the right of the summary table as `F-statistic`. The P-value for the test is right below it. In this case an F statistic of 114.0 yields a P-value of 1.70×10^{-17}, which is very small. Hence, we reject the null hypothesis, and conclude that this linear model fits the data well.

There is a lot more to these kinds of tests, which become more important in the multiple regression case. We'll come back to that later on!

Inference on Model Parameters

Now that we know how to find the line of best fit, we want to know how good our estimates of the slope and intercept are. We began by assuming that there is a linear model describing the relationship between x and Y:

$$Y = \alpha + \beta x + \varepsilon. \tag{6.21}$$

Because we only have limited data on the relationship between these two variables x and Y, we can't calculate α and β exactly, but only estimate them with our least squares estimates a and b. How can we know how good these estimates actually are? We turn again to our statistical inference tools of the confidence interval and hypothesis test.

A big question in linear regression is whether knowing the value of x makes a difference in our estimation of Y. If it doesn't, the slope value in the model would actually be 0. If it does, it would be something different from 0. But β is a parameter of the model, which we estimated by calculating the slope b using least squares. So we can use our hypothesis testing skills to decide if $\beta = 0$ or not!

$$H_0 : \beta = 0, H_1 : \beta \neq 0. \tag{6.22}$$

We can show that the test statistic for this hypothesis test is a T random variable, and its value is given in the model summary readout. Look for the column `t`, just two columns to the right of where you found the coefficients. And just next to that is the P-value for the test.

In this case, we got $T = 10.677$ which yields a P-value of 0.000. This causes us to reject the null hypothesis and conclude that the true slope is actually different from 0. We can also give a 95% confidence interval for our estimate of the slope, which is again given in the table as $(0.701, 1.022)$.

If this seems similar to the goodness of fit test we just covered, you're right! In fact, they are exactly the same in the simple linear regression case (verify for yourself that $F = T^2$). However, once we move to the multiple regression case, they will no longer be the same, and both tests will be important!

Before moving on, take some time to look over the model summary output again, and make sure that you can find all the information we've talked about so far.

Model Diagnostics

To wrap up our discussion of simple linear regression, we need to address a few details that we have glossed over up to this point. In our previous discussion, we have made a few assumptions which we need to take a deeper look at.

We assumed:

1. Y and x really do have a linear relationship. We made this assumption when we decided to use the model $Y = \alpha + \beta x + \varepsilon$.
2. The random error ε is normally distributed with mean 0.
3. The random error ε is independent of x.

These last two assumptions were not explicitly stated, but they are needed to make the regression interpretations work. And when we make assumptions like this, we need to check to see if these are reasonable! Here, we will show how to visually evaluate these assumptions using graphs.

The first question of the relationship between Y and x can be easily evaluated through the *scatter plot* we made at the beginning of this section. If the scatter plot isn't clear, then the correlation coefficient can help!

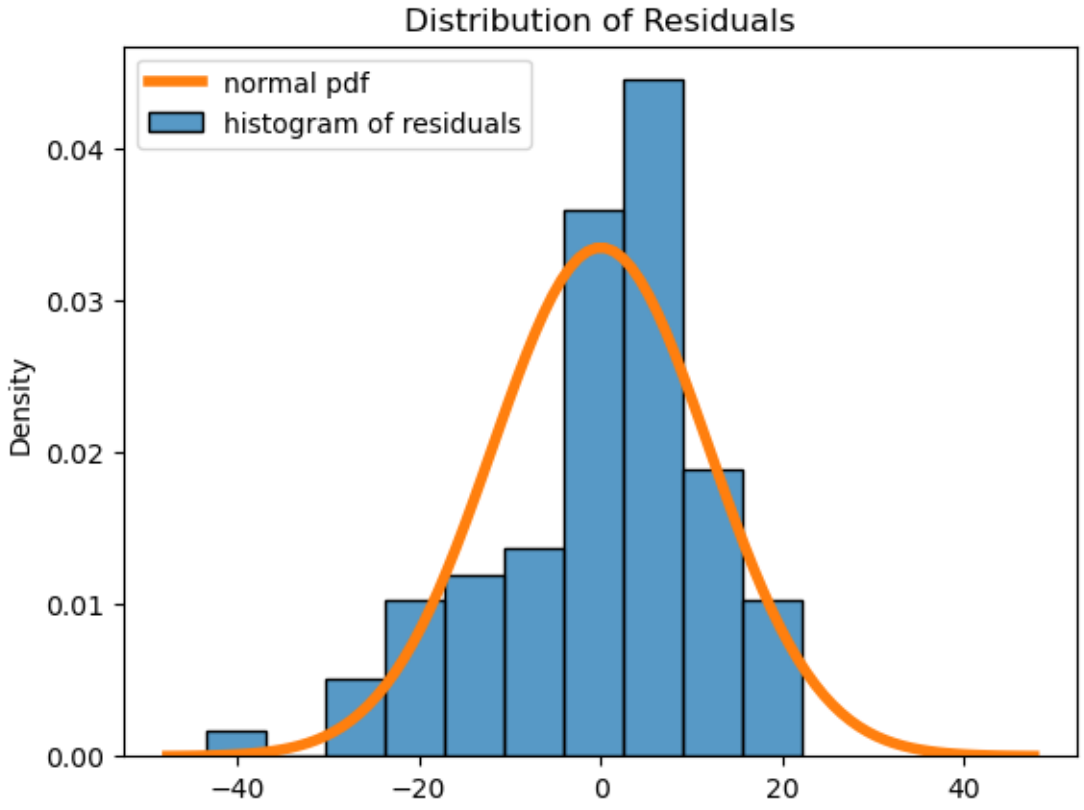
We can evaluate the next two claims by examining the residuals. The residuals are related to the random error since, according to the model,

$$y_i - \hat{y}_i = y_i - (a + bx_i) \approx y_i - (\alpha + \beta x_i) = \varepsilon_i. \tag{6.23}$$

Plotting the residuals will give us an idea about the distribution of the ε, but we need two different types of plots to examine the assumptions.

First, we can plot a histogram of the residuals:

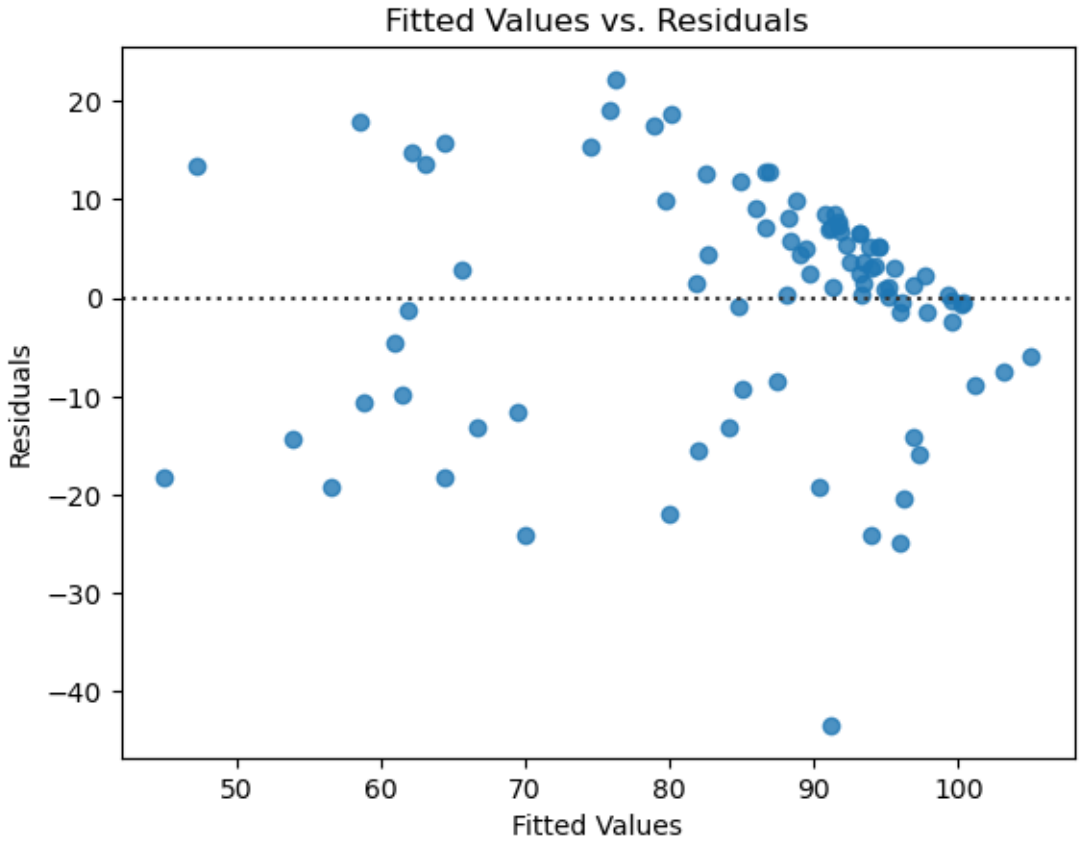
```
palette = iter(sns.color_palette())
sns.histplot(
    x = results.resid,
    stat = 'density',
    color = next(palette),
    label = "histogram of residuals",
)
std = np.std(results.resid)
x = np.linspace(-std*4, std*4, 100)
plt.plot(
    x,
    scipy.stats.norm.pdf(x, loc=np.mean(results.resid), scale=std),
    color = next(palette),
    linewidth = 4,
    label = "normal pdf"
)
plt.title("Distribution of Residuals")
plt.legend();
```



Here we see that the residuals have a roughly mound-shaped distribution, though there is left skew and the histogram does deviate somewhat from the normal distribution. We will say that the normal residual assumption is ok in this case, even though the residuals don't follow the normal distribution as much as we might like!

Second, we look at a scatter plot between x and the residuals. If the errors are truly independent of x, then we will not observe any patterns in the graph, and it should look like a random cloud of points. There should be roughly an equal number spread out above and below 0, with most of them being close to 0.

```
ax = sns.residplot(
    x = results.fittedvalues,
    y = results.resid,
)
ax.set_xlabel('Fitted Values')
ax.set_ylabel('Residuals')
ax.set_title("Fitted Values vs. Residuals");
```



Here there isn't a clear pattern, though the cluster of points in the upper right might give us some pause. Again it looks like the assumptions are reasonable, though that cluster means we should be cautious in this case about the fit of the model.

It can take some time to develop intuition about which plots look "good enough" and which are really bad, and there are some more sophisticated tests for these things that are outside the scope of this book. To illustrate the point, we provide an example below of what would it look like if the assumptions were badly violated, using a different variable from our data for the x-axis. Here is the model using this other variable:

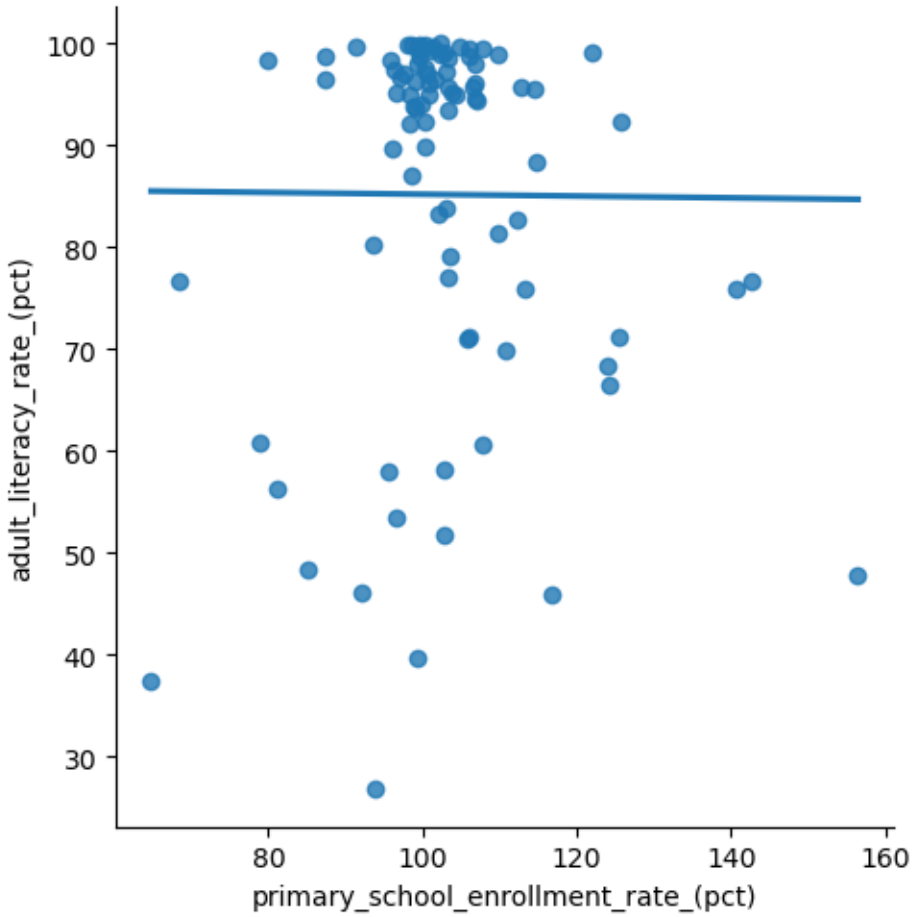
```
x = 'primary_school_enrollment_rate_(pct)'
y = 'adult_literacy_rate_(pct)'
model = sm.regression.linear_model.OLS(wbl[y], sm.tools.tools.add_constant(wbl[x]))
results = model.fit()
```

(continues on next page)

(continued from previous page)

```
print(results.summary())

sns.lmplot(
    data = wbl,
    x = x,
    y = y,
    ci = None,
);
```

```
                                OLS Regression Results
==============================================================================
Dep. Variable:     adult_literacy_rate_(pct)   R-squared:                       0.
 ↪000
Model:                                   OLS   Adj. R-squared:                 -0.
 ↪011
Method:                        Least Squares   F-statistic:                    0.
 ↪003546
Date:                       Thu, 20 Jul 2023   Prob (F-statistic):              0.
 ↪953
Time:                               16:47:39   Log-Likelihood:                -384.
 ↪08
No. Observations:                         89   AIC:                             ␣
 ↪772.2
Df Residuals:                             87   BIC:                             ␣
 ↪777.1
Df Model:                                  1
Covariance Type:                   nonrobust
==============================================================================================================
                                         coef    std err          t      P>|t|  ␣
 ↪   [0.025      0.975]
--------------------------------------------------------------------------------
 ↪---------------------
const                                  86.0796     15.388      5.594      0.000  ␣
 ↪   55.495     116.664
primary_school_enrollment_rate_(pct)    -0.0088      0.148     -0.060      0.953  ␣
 ↪   -0.303       0.286
==============================================================================
Omnibus:                       22.660   Durbin-Watson:                   1.728
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               29.654
Skew:                          -1.356   Prob(JB):                     3.64e-07
Kurtosis:                       3.800   Cond. No.                         823.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly␣
 ↪specified.
```



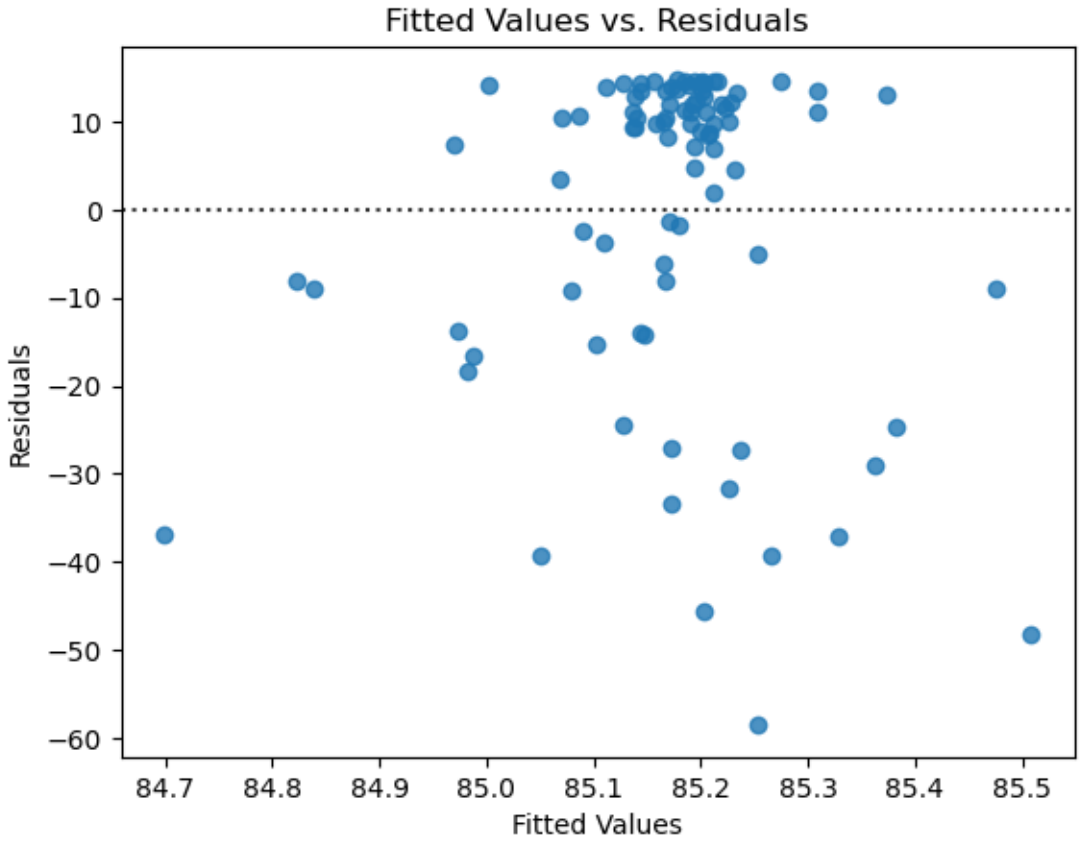
And here are the model diagnostic plots for this other model. Notice that the spread of the residuals is skewed and that the distribution does not look normal, violating our regression assumptions.

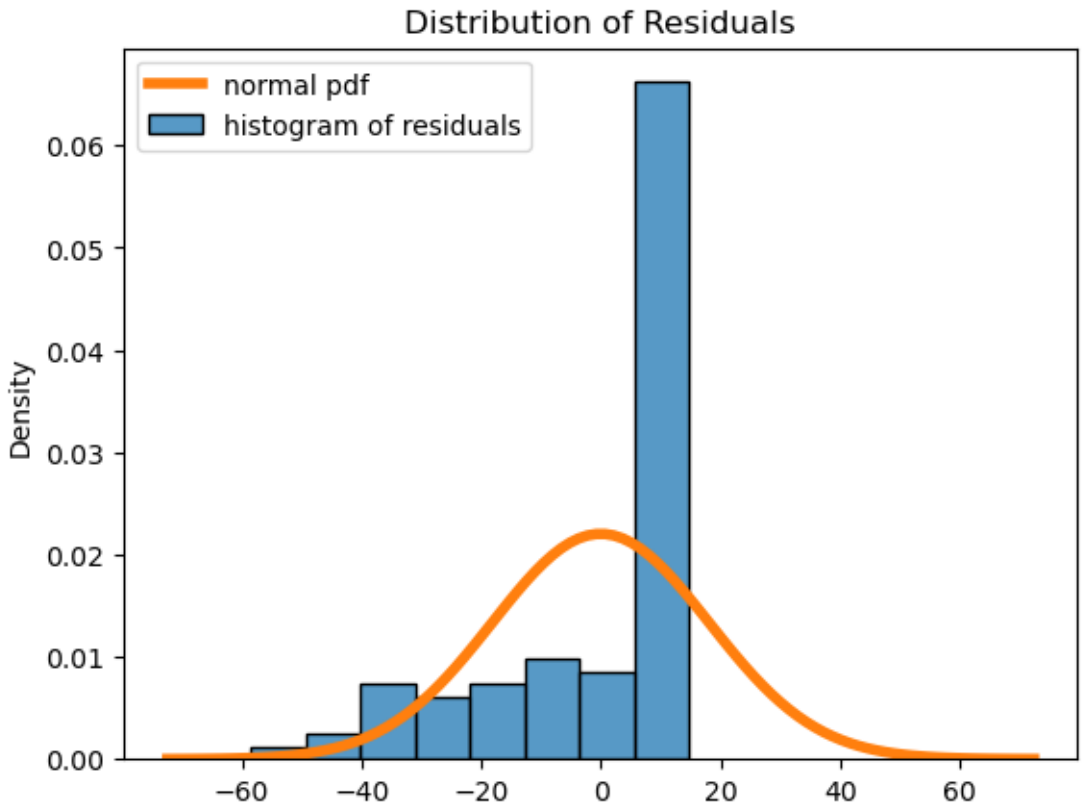
```
# plot residuals scatterplot
fig, ax = plt.subplots()
sns.residplot(
    x = results.fittedvalues,
    y = results.resid,
    ax = ax,
)
ax.set_xlabel('Fitted Values')
ax.set_ylabel('Residuals')
ax.set_title("Fitted Values vs. Residuals");

# plot residuals histogram
fig, ax = plt.subplots()
palette = iter(sns.color_palette())
sns.histplot(
    x = results.resid,
    stat = 'density',
    color = next(palette),
    label = "histogram of residuals",
)
std = np.std(results.resid)
x = np.linspace(-std*4, std*4, 100)
```

(continues on next page)

(continued from previous page)

```
plt.plot(
    x,
    scipy.stats.norm.pdf(x, loc=np.mean(results.resid), scale=std),
    color = next(palette),
    linewidth = 4,
    label = "normal pdf"
)
plt.title("Distribution of Residuals")
plt.legend();
```





In summary, if all of the diagnostic plots look ok, then the assumptions are reasonable, and our modeling results can be trusted! If not, there are other things you can try, but that is outside the scope of this section. It can take some time to get the hang of looking at these plots, but the more you practice the easier it becomes.

Exercise 1

What sorts of team statistics are associated with winning in the English Premier League? We can explore this question at a high level using simple linear regression.

Below, we load a dataset from the file `epl_table_22_23.csv` (click here to download a copy). This dataset contains the league standings along with some common statistics for the 2022-23 EPL season. A few notes:

- `points` is the total number of points accrued by a team through winning or drawing matches. A team earns 1 point for a draw and 3 points for a win.
- `goals_for` is the number of goals scored by the team, while `goals_against` is the number of goals allowed by the team. `goal_diff` is how many more (or less) goals a team scored than allowed.

```
epl = pd.read_csv('epl_table_22_23.csv')
epl.head()
```

```
          team  wins  draws  losses  points  goals_for  goals_against  \
0      Arsenal    26      6       6      84         88             43
1  Aston Villa    18      7      13      61         51             46
2  Bournemouth    11      6      21      39         37             71
3    Brentford    15     14       9      59         58             46
4     Brighton    18      8      12      62         72             53
```

(continues on next page)

(continued from previous page)

```
   goal_diff  expected_goals_for  expected_goals_against  expected_goal_diff  \
0         45                71.9                    42.0                29.9
1          5                50.2                    52.5                -2.2
2        -34                38.6                    63.9               -25.3
3         12                56.8                    49.9                 6.8
4         19                73.3                    50.2                23.1

   attendance
0       60191
1       39485
2       10362
3       17078
4       31477
```

1. Make a scatter plot of the number of goals a team scored (`goals_for`) versus the number of points they accrued in the standings (`points`). How would you describe the relationship?
2. Fit a linear model to these two variables, and print out the summary.
3. Use the summary to write down the estimated slope and intercept of the model. Interpret these values in the context of the problem. How many points, on average, is each additional goal scored in the standings worth?
4. What is the value of r^2? Does the model seem to be a good fit? Explain your reasoning.
5. Provide diagnostic plots for the model and comment on the validity of the assumptions.
6. Repeat this analysis with another variable or variables. Do your new models seem to be a better or worse fit? Why or why not? Provide diagnostic plots for your model and comment on the validity of the assumptions.
7. Do you think `attendance` has any effect on a team's performance? Use some of the tools from this section to answer, and explain what you find. How might you use this example to help understand the common phrase "correlation is not causation"?

6.5.2 Multiple Regression

Much more interesting is the case where there are several independent variables that are all possibly connected to an output. In this case, because we are dealing with multiple variables, our modeling tool is now called **multiple regression**.

For our example, we will return to the Palmer Penguins dataset introduced in the *previous section*. Specifically, we will look at the connection between body mass and some of the other measurements of the penguins.

This becomes a bit more difficult to visualize since we are dealing with more than just 2 variable dimensions.

```
from palmerpenguins import load_penguins

penguins = load_penguins()
penguins.sample(5).head() # display 5 randomly sampled rows
```

```
     species     island  bill_length_mm  bill_depth_mm  flipper_length_mm  \
52    Adelie     Biscoe            35.0           17.9              190.0
109   Adelie     Biscoe            43.2           19.0              197.0
214   Gentoo     Biscoe            45.7           13.9              214.0
199   Gentoo     Biscoe            50.5           15.9              225.0
128   Adelie  Torgersen            39.0           17.1              191.0

     body_mass_g     sex  year
52        3450.0  female  2008
```

(continues on next page)

(continued from previous page)

```
109       4775.0    male  2009
214       4400.0  female  2008
199       5400.0    male  2008
128       3050.0  female  2009
```

So what does it look like to understand how body mass changes based on these other variables? Mathematically, we are modeling Y as a linear function of several independent variables $x_1, x_2, \ldots, x_p$. As in the single variable case, our variable Y won't follow this linear function exactly. Instead, it will be subject to some random deviations, modelled by ε.

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p + \varepsilon. \tag{6.24}$$

Again, we only have limited data, so we need to *estimate* the values of $\beta_0, \ldots, \beta_p$ with some estimators $b_0, \ldots, b_p$. As a result, our new equation looks like this:

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_p x_p. \tag{6.25}$$

In this equation, b_0 is an intercept, while $b_1, \ldots, b_p$ are slope parameters associated with the different variables $x_1, \ldots, x_p$. For example, the value of b_3 tells us how much we estimate that Y would change when x_3 changes, *with all other independent variables held constant*. Thinking about this in the abstract setting can be difficult, so we recommend coming back to this later in light of the example using real data below.

So how do we find this 'best' linear function? The good news is that we can do this estimation using least squares! Similar to the one-variable case, this involves finding the values of $b_0, \ldots, b_p$ that minimize the sum of the squared residuals:

$$\min_{b_0,\ldots,b_p} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{6.26}$$

We'll spare you the details in this section, as the notation can get messy and a full solution requires the use of linear algebra. You can find the details in most any college linear algebra book, including *this section* of this book.

Again, we have some code that can take care of this least squares minimization for us and find the estimates we need. As before, we'll use the `statsmodels` library, but when we specify the input variables `x`, we list each of the variables we are using to try to understand body mass. In this case, we start by looking at just bill depth and flipper length.

Note that we also add `missing = 'drop'` since a few of these data points are missing, which means we need to tell the computer to ignore those rows!

```
x = penguins[['bill_depth_mm', 'flipper_length_mm']]
y = penguins['body_mass_g']
model1 = sm.regression.linear_model.OLS(y, sm.tools.tools.add_constant(x), missing=
↪'drop')
results = model1.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            body_mass_g   R-squared:                       0.761
Model:                            OLS   Adj. R-squared:                  0.760
Method:                 Least Squares   F-statistic:                     539.8
Date:                Thu, 20 Jul 2023   Prob (F-statistic):          4.23e-106
Time:                        16:47:40   Log-Likelihood:                -2527.0
No. Observations:                 342   AIC:                             5060.
Df Residuals:                     339   BIC:                             5071.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
```

(continues on next page)

(continued from previous page)

```
                         coef    std err          t      P>|t|      [0.025      0.
↪975]
------------------------------------------------------------------------------
↪--
const              -6541.9075    540.751    -12.098      0.000   -7605.557   -5478.
↪258
bill_depth_mm         22.6341     13.280      1.704      0.089      -3.488      48.
↪756
flipper_length_mm     51.5414      1.865     27.635      0.000      47.873      55.
↪210
==============================================================================
Omnibus:                        5.490   Durbin-Watson:                   2.069
Prob(Omnibus):                  0.064   Jarque-Bera (JB):                5.361
Skew:                           0.305   Prob(JB):                       0.0685
Kurtosis:                       3.067   Cond. No.                     5.14e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly␣
↪specified.
[2] The condition number is large, 5.14e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Again, there is a lot of information here! First, we can take the coefficients to see that the estimated relationship between body mass (Y) and the bill depth (x_1) and flipper length (x_2) is

$$\hat{y} = -6541.91 + 22.63x_1 + 51.54x_2 \tag{6.27}$$

This means that, on average, an increase of 1mm of bill depth is associated with an increase of about 23g in mass, at least for two penguins with the same flipper length. Likewise, holding bill depth constant, an increase of 1mm in flipper length is associated with an increase of about 52g in mass.

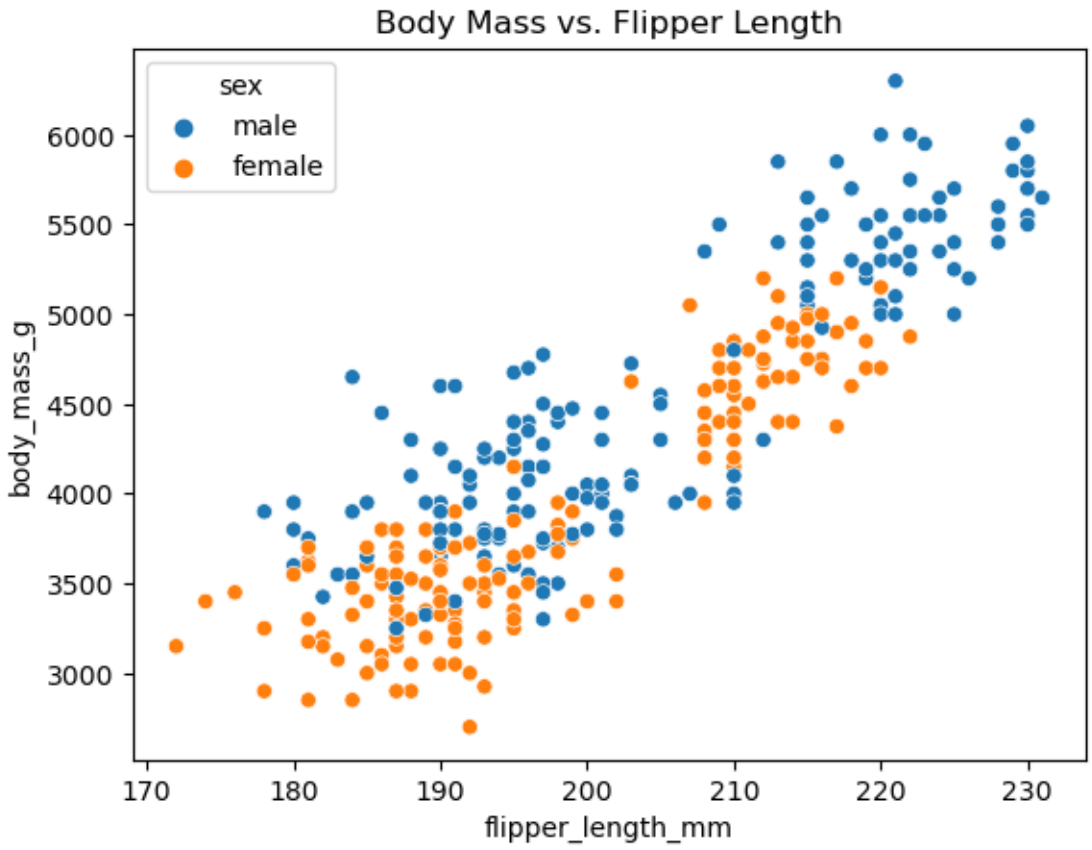
However, if we take a look at the P-values (in the `P>|t|` column), we see that the coefficient for `bill_depth_mm` has a value of 0.089. This is a somewhat large P-value, which indicates that the variable `bill_depth_mm` does not have a significant effect on `body_mass_g` in the presence of the other variable, `flipper_length_mm`. In other words, `bill_depth_mm` does not add any additional information beyond what `flipper_length_mm` already offers to our model for body mass. (See what happens when you exclude bill depth from the model!)

At this point, we have only one variable that seems to be associated with body mass. However, there is more we can do here. (And don't worry, I haven't forgotten about model assumptions and diagnostics—we'll come back to that soon!)

Dummy Variables

Categorical or qualitative variables can sometimes also be incorporated into a linear regression model by an appropriate transformation. In this case, we have at least one such variable that might be helpful in understanding body mass. Male penguins are typically larger than female penguins, so it might be helpful to incorporate the `sex` of the penguin into our model.

```
ax = sns.scatterplot(
    data = penguins,
    x = 'flipper_length_mm',
    y = 'body_mass_g',
    hue='sex'
)
ax.set_title("Body Mass vs. Flipper Length");
```



However, this variable doesn't consist of numerical values. Instead, it consists of a set of categorical values (`male` and `female` in this case), which are referred to as the "levels" of the variable. So it's not clear how we can incorporate this variable into the linear model as-is.

To solve this problem, we will "encode" the levels of this categorical variable as numbers. This process is called adding a "dummy" variable. To do this, we'll use the `get_dummies` function from the `pandas` library:

```
x = penguins[['sex', 'flipper_length_mm']]
y = penguins['body_mass_g']

x = pd.get_dummies(x, drop_first="TRUE", dtype=int)
x.head(10)
```

	flipper_length_mm	sex_male
0	181.0	1
1	186.0	0
2	195.0	0
3	NaN	0
4	193.0	0
5	190.0	1
6	181.0	0
7	195.0	1
8	193.0	0
9	190.0	0

In this case, we see that we have created a new dummy variable `sex_male`. This is a variable that takes the value 1 if the sex of the penguin is male and 0 otherwise. (There is nothing special about whether male or female is 0 or 1; the software chooses automatically based on which levels appear first in the data.)

So now we have turned our categorical variable into a numerical one, which we can add to the model. Let's see what it looks like!

```
model2 = sm.regression.linear_model.OLS(y, sm.tools.tools.add_constant(x), missing=
↪'drop')
results = model2.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            body_mass_g   R-squared:                       0.803
Model:                            OLS   Adj. R-squared:                  0.802
Method:                 Least Squares   F-statistic:                     691.0
Date:                Thu, 20 Jul 2023   Prob (F-statistic):          2.53e-120
Time:                        16:47:41   Log-Likelihood:                -2493.9
No. Observations:                 342   AIC:                             4994.
Df Residuals:                     339   BIC:                             5005.
Df Model:                           2
Covariance Type:            nonrobust
================================================================================================
                       coef    std err          t      P>|t|      [0.025      0.
↪975]
------------------------------------------------------------------------------------------------
↪--
const             -5327.2009    281.739    -18.908      0.000   -5881.378   -4773.
↪024
flipper_length_mm    46.5787      1.420     32.795      0.000      43.785      49.
↪372
sex_male            347.2632     39.891      8.705      0.000     268.797     425.
↪729
==============================================================================
Omnibus:                        0.303   Durbin-Watson:                   1.744
Prob(Omnibus):                  0.860   Jarque-Bera (JB):                0.412
Skew:                           0.057   Prob(JB):                        0.814
Kurtosis:                       2.874   Cond. No.                     2.94e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly␣
↪specified.
[2] The condition number is large, 2.94e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Take a look at the output of this new model. One way to think about this is that there are two lines, one for each category (male and female). If the penguin is male ($x_2 = 1$), the regression line looks like

$$\hat{y}_{male} = -5327.20 + 46.58x_1 + 347.26x_2 = -5327.20 + 46.58x_1 + 347.26(1) = -4979.94 + 46.58x_1. \quad (6.28)$$

For a female penguin, on the other hand, the variable x_2 would take a value of 0. See if you can write down the estimated line for a female penguin.

You can also simply interpret the coefficient of `sex_male` to mean that male penguins are on average 347.26g heavier than female penguins when controlling for bill depth.

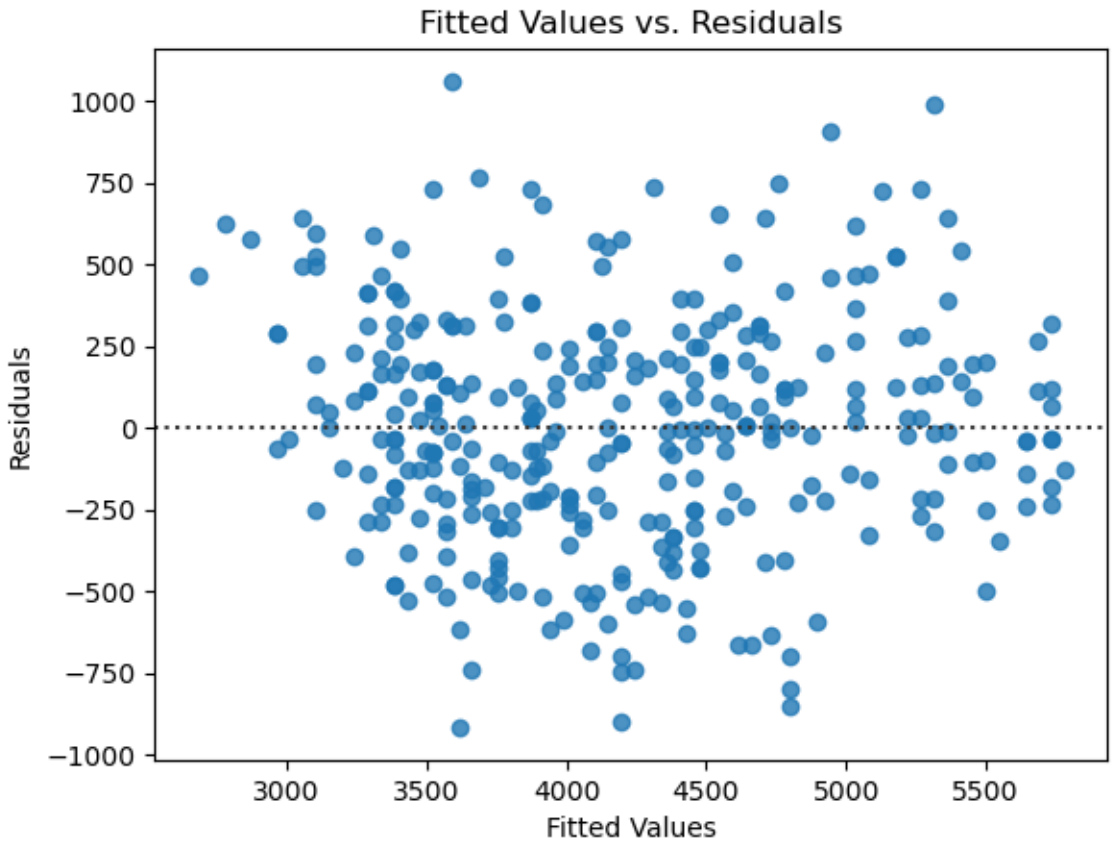
Notice that we only needed 1 dummy variable to describe a categorical variable with two possible levels. In general, we need to create one fewer variable than the number of levels in the variable. This is what the `drop_first=TRUE` argument does: it drops one of the dummies that isn't necessary in the model. In the exercises, you'll see what happens when we add in a categorical variable with 3 levels (`species`, which has levels `Gentoo`, `Adelie`, or `Chinstrap`).

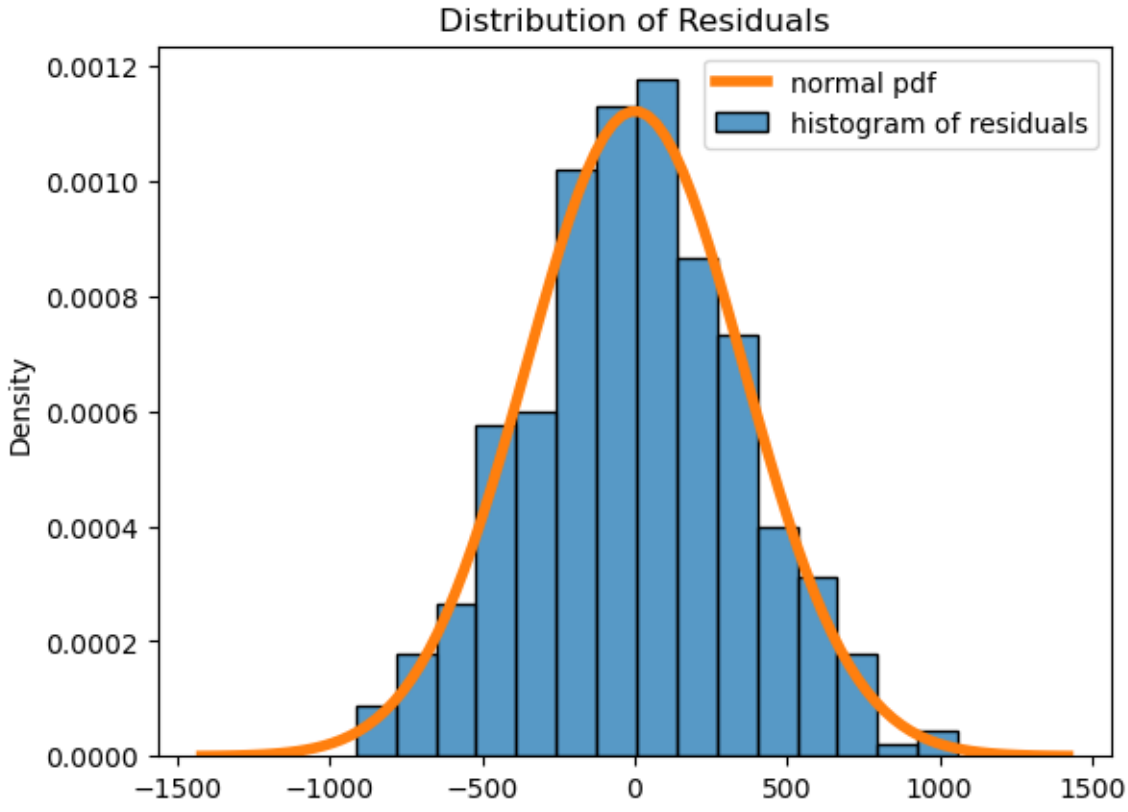
Model Diagnostics

The assumptions for a multiple linear regression model are nearly the same as for a simple linear regression model, and they are not much more difficult to evaluate. Once again, we need to assume that the relationship between Y and the input variables $x_1, \ldots, x_p$ is linear. And as before, we can check this assumption by looking at the *scatter plot* we made.

We also need to assume that the residuals are nearly normal and independent of the input variables. Coding the plots to check these last two assumptions looks much the same as it did before.

```
# plot residuals scatterplot
fig, ax = plt.subplots()
sns.residplot(
    x = results.fittedvalues,
    y = results.resid,
    ax = ax,
)
ax.set_xlabel('Fitted Values')
ax.set_ylabel('Residuals')
ax.set_title("Fitted Values vs. Residuals")

# plot residuals histogram
fig, ax = plt.subplots()
palette = iter(sns.color_palette())
sns.histplot(
    x = results.resid,
    stat = 'density',
    color = next(palette),
    label = "histogram of residuals",
)
std = np.std(results.resid)
x = np.linspace(-std*4, std*4, 100)
plt.plot(
    x,
    scipy.stats.norm.pdf(x, loc=np.mean(results.resid), scale=std),
    color = next(palette),
    linewidth = 4,
    label = "normal pdf"
)
plt.title("Distribution of Residuals")
plt.legend();
```


Fitted Values vs. Residuals
Residuals
Fitted Values
1000
750
500
250
0
−250
−500
−750
−1000
3000
3500
4000
4500
5000
5500


Distribution of Residuals
normal pdf
histogram of residuals
Density
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
−1500
−1000
−500
0
500
1000
1500

It's pretty clear that the residuals are nearly normal and that there is no pattern. In fact, this looks much better than the plots of the residuals for the simple linear regression models, indicating that in some respects the linear model is a better fit here.

But we've glossed over something important! If you looked closely at the *original scatter plot*, you probably noticed that male penguins don't just have larger body mass, they also have larger flipper length. In other words, the input variables `sex` and `flipper_length_mm` are themselves correlated, which could be causing an issue in our model. (Note that the model readout also indicates that this might be a problem.) We can get a better look at this through some *exploratory data analysis*. But for now, let's move on to some other topics.

Inference on Model Parameters

The techniques of inference that we talked about in the single-variable case carry over to the multiple regression model. The main difference is the interpretation of the readout results. Because we have multiple variables involved in the model, which themselves may be linearly connected, the matter becomes slightly more complex.

First, we might ask how well the model is doing overall. How well does it fit the data? The software has again performed the *Goodness of Fit Test* for us, comparing (in basic terms) the null hypothesis that the model is not a good fit versus the alternative hypothesis that it is a good fit. Looking back at the model readout, we can find that our F-statistic is 691.0, yielding an extremely small P-value of 2.53×10^{-120} (0, for our purposes). This means that our model is a good fit and that our input variables are providing some information about `body_mass_g`.

This test looks at the fit of the model all at once, but we can also look in more detail at the values of the individual "slope" coefficients associated with each input. The model readout provides a separate hypothesis test for whether each coefficient is significantly different from zero in the columns `t` (the test statistic) and `P>|t|` (the P-value). Moreover, the readout also gives 95% confidence intervals for each coefficient. This information can help you determine the effect that a single variable (such as flipper length) has on body mass when you also account for the differences in the other variables (such as the sex of the animal).

There is a lot more to say about this, and about regression as a whole, but now you should have enough knowledge of the basics to be able to build and interpret a regression model on your own!

Exercise 2

Create a new model using `sex`, `species`, and `flipper_length_mm` to predict `body_mass_g`.

1. How many different levels (i.e., possible categories) are in the `species` variable? How does the software handle the `species` variable?
2. Are all the variables providing useful information in your model? How can you tell?
3. Describe the typical characteristics of a very large or very small penguin.
4. Provide diagnostics for your model and comment on the validity of the regression assumptions.

6.6 Solutions to Exercises

6.6.1 Foundations of Statistical Inference

Exercise 1

1. Use the widget above to construct 25 different 90% confidence intervals for the average weekly wages of Premier League players. How many of the 25 contain the true population mean? Is this the number you expect? Would it be possible for exactly 90% of the intervals to contain the true population mean?

I observed 23/25 = 96% of the 90% confidence intervals contained the true average. (You may observe a different percentage because the random samples in the widget will be different each time the code is run.) It would not be possible to observe exactly 95% since 95% of 25 is 22.5.

2. Choose your own confidence level, and construct a single confidence interval for the average weekly wages of a Premier League player. Describe your interval in a complete sentence.

I constructed a 98% confidence interval on a sample of 45 players. Since this confidence level is high, the interval was quite wide.

I have 98% confidence that the true average weekly salary is between 48366.561 and 110522.328 GBP.

6.6.2 Hypothesis Testing for Categorical Data

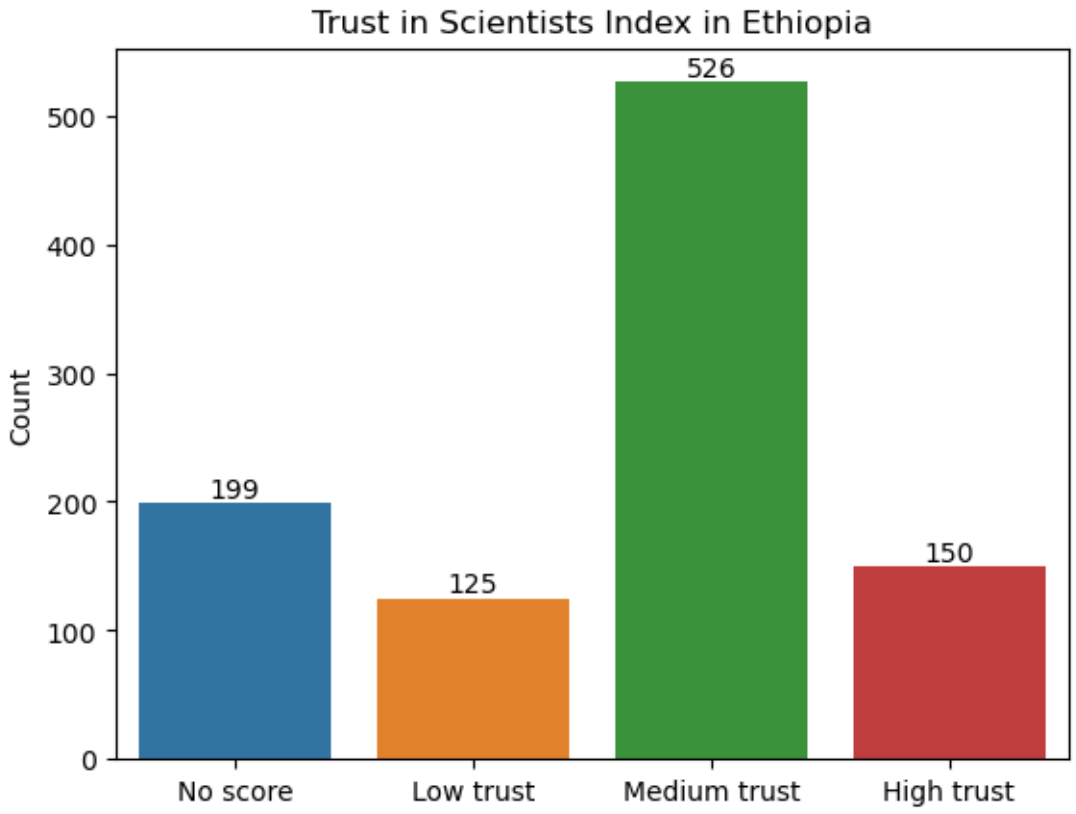
Exercise 1

1. Pick another country (perhaps your country of origin or another country you are interested in) and filter the observations to just include responses from that country.

For this sample solution, we have chosen to look at trust in scientists in Ethiopia. (To pick a different country, change the value of the variable `country1`.)

After loading in the Wellcome Global Monitor survey data (click here to download a copy), we create a new dataframe called `wgm_country1` that includes only the observations for Ethiopia.

```
country1 = "Ethiopia"
wgm = pd.read_csv("wellcome_global_monitor_2018.csv")
wgm_country1 = wgm[wgm["Country"] == country1]
```



2. Make a hypothesis about the proportion of all the people in your chosen country that have high trust in scientists. For example, do you think it is over 20%? Under 15%? Make any hypothesis you like.

Our null hypothesis is that the proportion of people with high trust in scientists in Ethiopia is 15%. Our alternative hypothesis is that the proportion is not 15%. Note that this is a two-sided hypothesis.

$$H_0 : p = 0.15, H_1 : p \neq 0.15. \tag{6.29}$$

3. Choose a level of α.

We will choose to set $\alpha = 0.1$.

```
alpha = 0.1
```

4. Compute the value of $\hat{p}$.

Here is the code to calculate $\hat{p}$.

```
# number of successes (i.e., people with high trust in scientists)
count = len(wgm_country1[wgm_country1["Trust_Index"] == "High trust"])

# sample size (i.e., the total number of observations)
nobs = len(wgm_country1["Trust_Index"])

# compute sample proportion
p_hat =  count / nobs
print(f"The sample proportion is {p_hat:.3f}. There are {nobs} total observations.")
```

```
The sample proportion is 0.150. There are 1000 total observations.
```

The sample proportion is 15%, which seems to support our null hypothesis. However, this isn't enough evidence to make a conclusion yet, since the sample proportion is variable and won't be exactly the same as the population proportion. We need to conduct a hypothesis test before we can make our conclusion.

5. Run a hypothesis test using `statsmodels`.

Here is the code to conduct the hypothesis test using `statsmodels`. We can use the same variables `count` and `nobs` that we defined earlier. Note that we set `alternative` to "two-sided" because our alternative hypothesis is that $p \neq 0.15$.

```
p0 = 0.15 # proportion under null hypothesis
(stat, pval) = sm.stats.proportion.proportions_ztest(
    count = count, # number of successes
    nobs = nobs, # number of observations
    value = p0,
    alternative = "two-sided",
    prop_var = p0, # use proportion under null hypothesis to calculate variance for␣
 ↪test statistic
)
print(f"Z-test for proportion: test statistic is {stat:.3f}, P-value is {pval:.3f}.")
```

```
Z-test for proportion: test statistic is 0.000, P-value is 1.000.
```

6. Write your conclusion in a complete sentence. Be sure to report the test statistic and the P-value. If you found a significant result, give a confidence interval for the proportion.

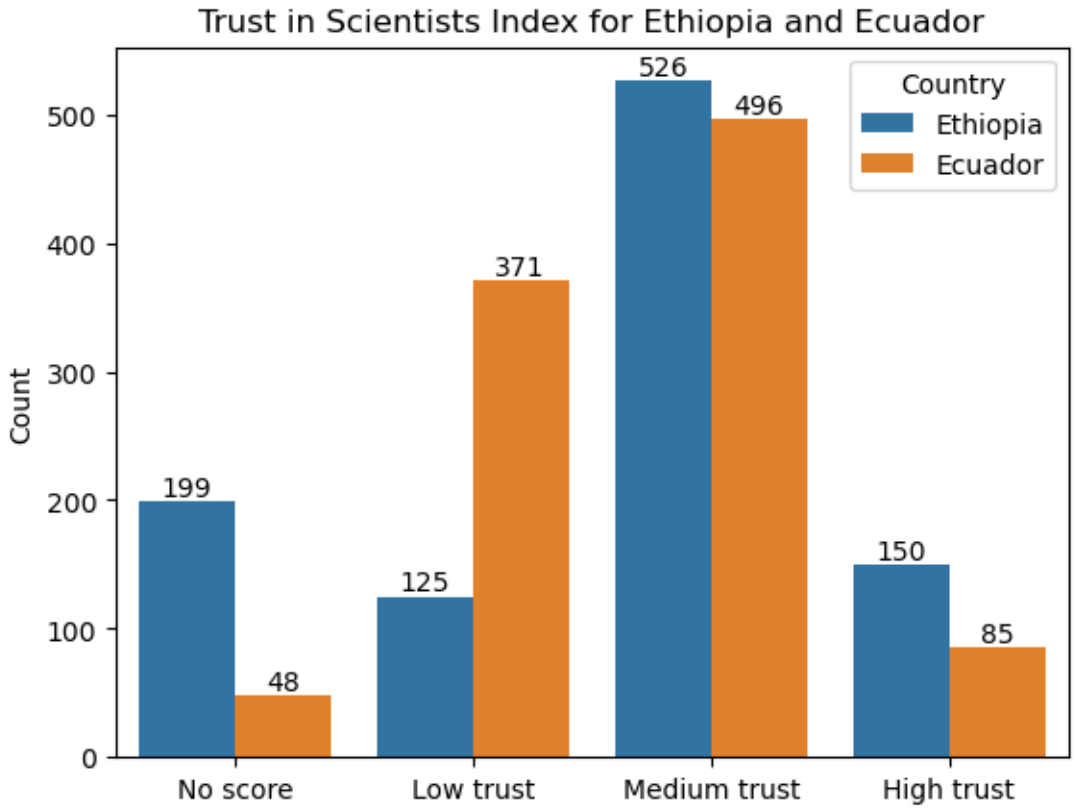
Our P-value is 1, which is greater than our threshold of $\alpha = 0.1$. So we fail to reject the null hypothesis that the proportion of people in Ethiopia with high trust in scientists is 15%.

Exercise 2

1. Pick a second country of interest.

For this sample solution, we have chosen Ecuador as the second country. (To pick a different country, change the value of the variable `country2`.) As before, we create a new dataframe called `wgm_country2` that includes only the observations for Ecuador.

```
country2 = "Ecuador"
wgm_country2 = wgm[wgm["Country"] == country2]
```



2. Write hypotheses to test whether the proportion of people in the first country with high trust in scientists is higher than the same proportion in the second country.

Our null hypothesis is that the proportion of people who have high trust in scientists in Ethiopia (p_1) is the same as the proportion of people who have high trust in scientists in Ecuador (p_2). Our alternative hypothesis is that the proportion of people who have high trust in scientists in Ethiopia is higher than the corresponding proportion in Ecuador ($p_1 > p_2$).

$$H_0 : p_1 - p_2 = 0, H_1 : p_1 - p_2 > 0. \tag{6.30}$$

(Recall that the null hypothesis should always be in the form "population parameter equals something". So when we are asked to hypothesize that the first proportion is higher than the second proportion, that means that it is our alternative hypothesis that is $p_1 - p_2 > 0$, not our null hypothesis.)

3. Choose a level of α.

As before, we will choose $\alpha = 0.1$.

```
alpha = 0.1
```

4. Run the hypothesis test using `statsmodels`.

Here is the code to conduct the hypothesis test using `statsmodels`. Note that we set `alternative` to "larger" because our alternative hypothesis is that $p_1 - p_2 > 0$.

```
# number of successes (i.e., people with high trust in scientists)
count1 = len(wgm_country1[wgm_country1["Trust_Index"] == "High trust"])
count2 = len(wgm_country2[wgm_country2["Trust_Index"] == "High trust"])

# sample size (i.e., the total number of observations)
nobs1 = len(wgm_country1["Trust_Index"])
nobs2 = len(wgm_country2["Trust_Index"])

# run the hypothesis test using statsmodels
p0 = 0 # proportion under null hypothesis
(stat, pval) = sm.stats.proportion.proportions_ztest(
    count = [
        count1, # number of successes for sample 1
        count2, # number of successes for sample 2
    ],
    nobs = [
        nobs1, # number of observations for sample 1
        nobs2, # number of observations for sample 2
    ],
    value = p0,
    alternative = "larger",
    prop_var = False, # use pooled sample proportion to calculate variance for test␣
 ↪statistic
)
print(f"Z-test for difference of two proportions: test statistic is {stat:.3f}, P-
 ↪value is {pval:.3f}.")
```

```
Z-test for difference of two proportions: test statistic is 4.514, P-value is 0.
 ↪000.
```

5. Write your conclusion in a complete sentence. Be sure to report the test statistic and the P-value. If you found a significant difference, give a confidence interval for the difference in proportions.

Our P-value is 0, which is smaller than our threshold of $\alpha = 0.1$. So we reject the null hypothesis in favor of the alternative hypothesis that the proportion of people in Ethiopia with high trust in scientists is larger than the corresponding proportion of people in Ecuador.

```
98% confidence interval is (42775.636, 90913.253).
```

6.6.3 Hypothesis Testing for Numerical Data

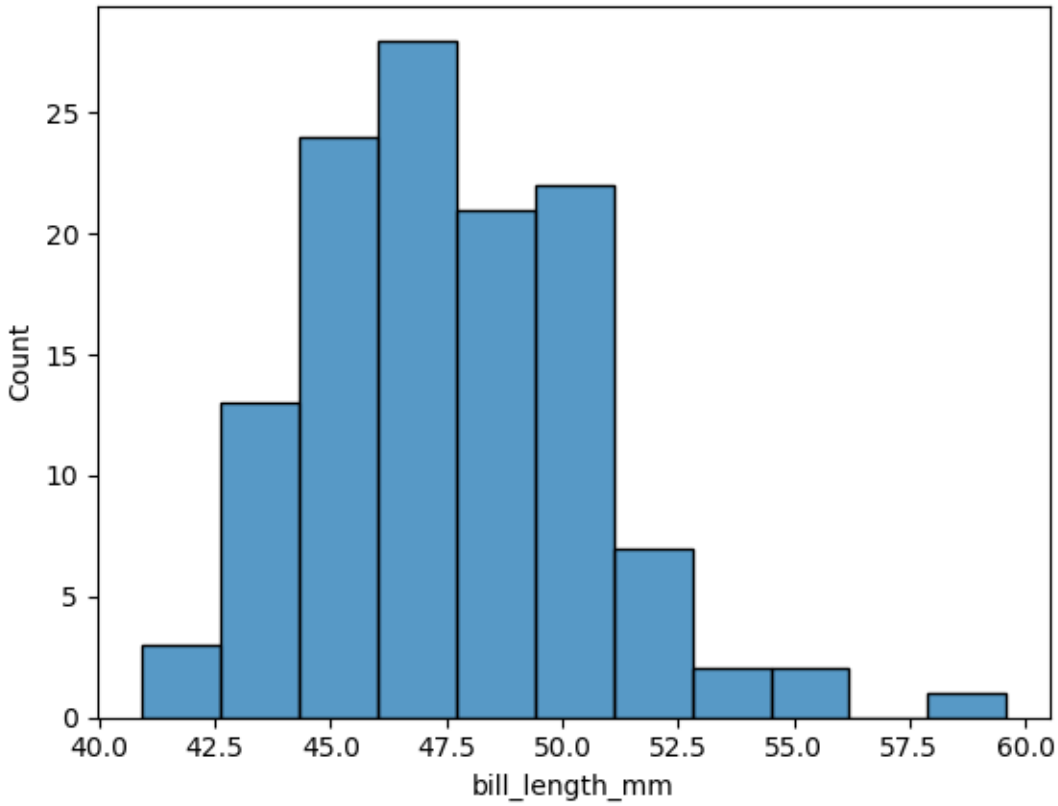
Exercise 1

1. Find the average bill length for Gentoo penguins from the Encyclopedia of Life. Use this information to form null and alternative hypotheses, making sure to note the units.

The average Gentoo penguin has a bill length of 53.8 mm.

2. What is the sample size you have? Does the distribution of bill lengths appear to meet the normality assumption needed to perform the T-test?

There are 992 elements in the dataframe. This is a large number, so we don't strictly need the normality assumption due to the CLT. However, the histogram shows that the normality assumption is reasonable.



3. Perform the test and report the test statistic and P-value.

```
T-test for mean: test statistic is -22.654 with 991 degrees of freedom.
P-value is 0.000.
```

4. What do you conclude? If you find a significant difference, give a confidence interval for average bill length.

The small P-value indicates that there is a significant difference in the bill lengths in the sample and the hypothesized mean of 53.8mm. A 95% confidence interval for the average bill length is 46.955 to 48.055 mm.

```
confidence_level = 0.95
(lower_ci, upper_ci) = result.confidence_interval(confidence_level) # use the result␣
↪object from the t-test
print(f"95% confidence interval is ({lower_ci:.3f}, {upper_ci:.3f}).")
```

```
95% confidence interval is (46.955, 48.055).
```

6.6.4 Linear Regression

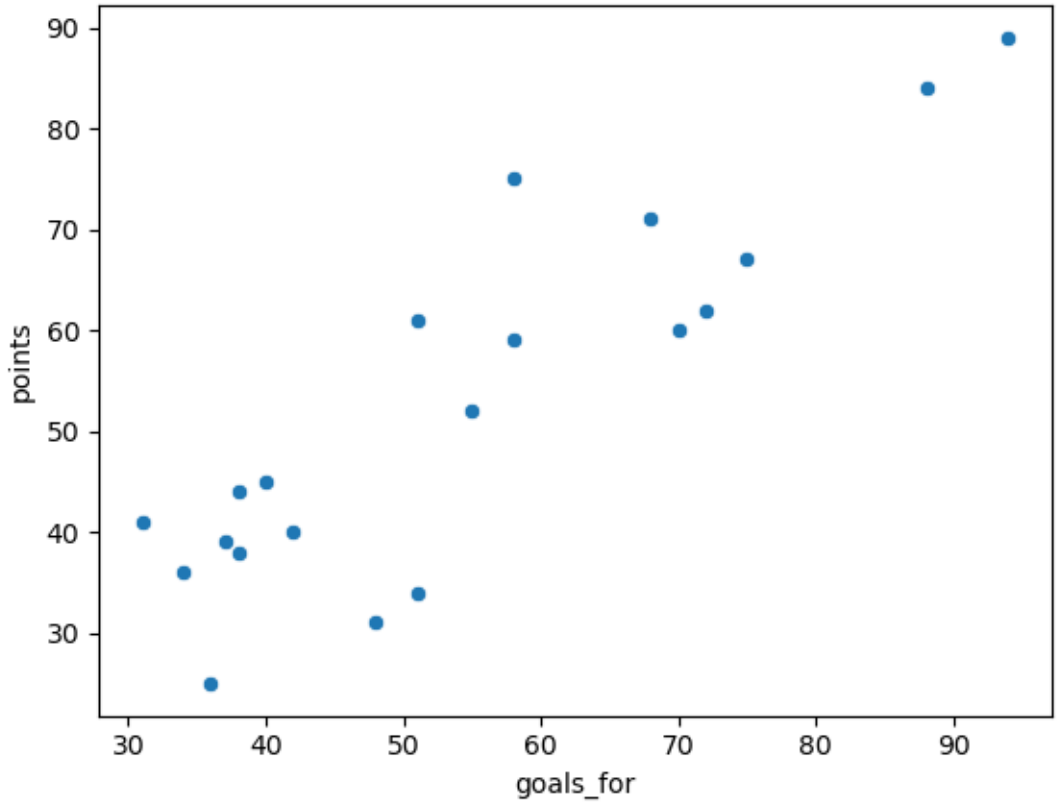
Exercise 1

```
epl = pd.read_csv('epl_table_22_23.csv')
epl.head()
```

```
          team  wins  draws  losses  points  goals_for  goals_against  \
0      Arsenal    26      6       6      84         88             43
1  Aston Villa    18      7      13      61         51             46
2  Bournemouth    11      6      21      39         37             71
3    Brentford    15     14       9      59         58             46
4     Brighton    18      8      12      62         72             53

   goal_diff  expected_goals_for  expected_goals_against  expected_goal_diff  \
0         45                71.9                    42.0                29.9
1          5                50.2                    52.5                -2.2
2        -34                38.6                    63.9               -25.3
3         12                56.8                    49.9                 6.8
4         19                73.3                    50.2                23.1

   attendance
0       60191
1       39485
2       10362
3       17078
4       31477
```

1. Make a scatter plot of the number of goals a team scored (`goals_for`) versus the number of points they accrued in the standings (`points`). How would you describe the relationship?

The relationship is positive, and somewhat linear.



2. Fit a linear model to these two variables, and print out the summary.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 points   R-squared:                       0.773
Model:                            OLS   Adj. R-squared:                  0.761
Method:                 Least Squares   F-statistic:                     61.44
Date:                Thu, 20 Jul 2023   Prob (F-statistic):           3.27e-07
Time:                        14:27:49   Log-Likelihood:                -71.037
No. Observations:                  20   AIC:                             146.1
Df Residuals:                      18   BIC:                             148.1
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.6042      6.323      0.886      0.387      -7.680      18.889
goals_for      0.8680      0.111      7.838      0.000       0.635       1.101
==============================================================================
Omnibus:                        0.519   Durbin-Watson:                   1.133
Prob(Omnibus):                  0.771   Jarque-Bera (JB):                0.030
```

(continues on next page)

(continued from previous page)

```
Skew:                          -0.087   Prob(JB):                         0.985
Kurtosis:                       3.074   Cond. No.                         182.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly␣
 ↪specified.
```

3. Use the summary to write down the estimated slope and intercept of the model. Interpret these values in the context of the problem. How many points, on average, is each additional goal scored in the standings worth?

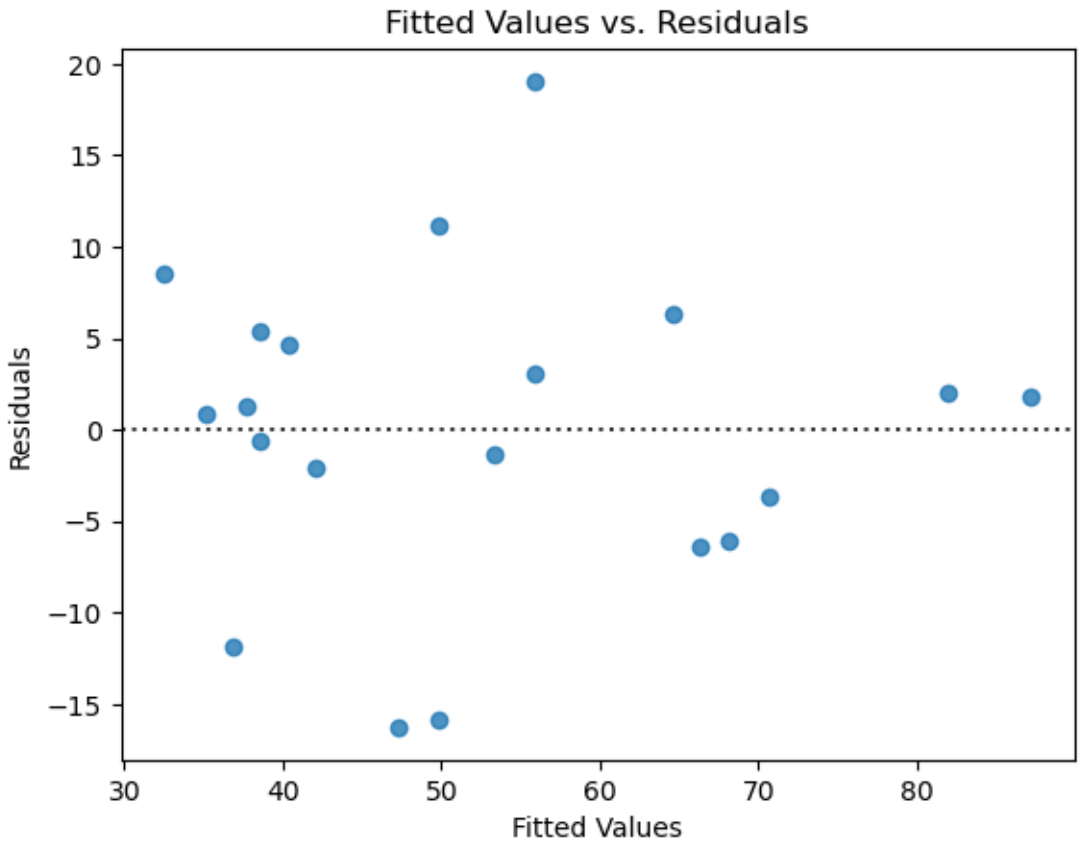
The summary gives an estimate line of $\hat{points} = 5.6042 + .868 * goalsfor$. This means that each goal is worth on average .868 points in the standings.

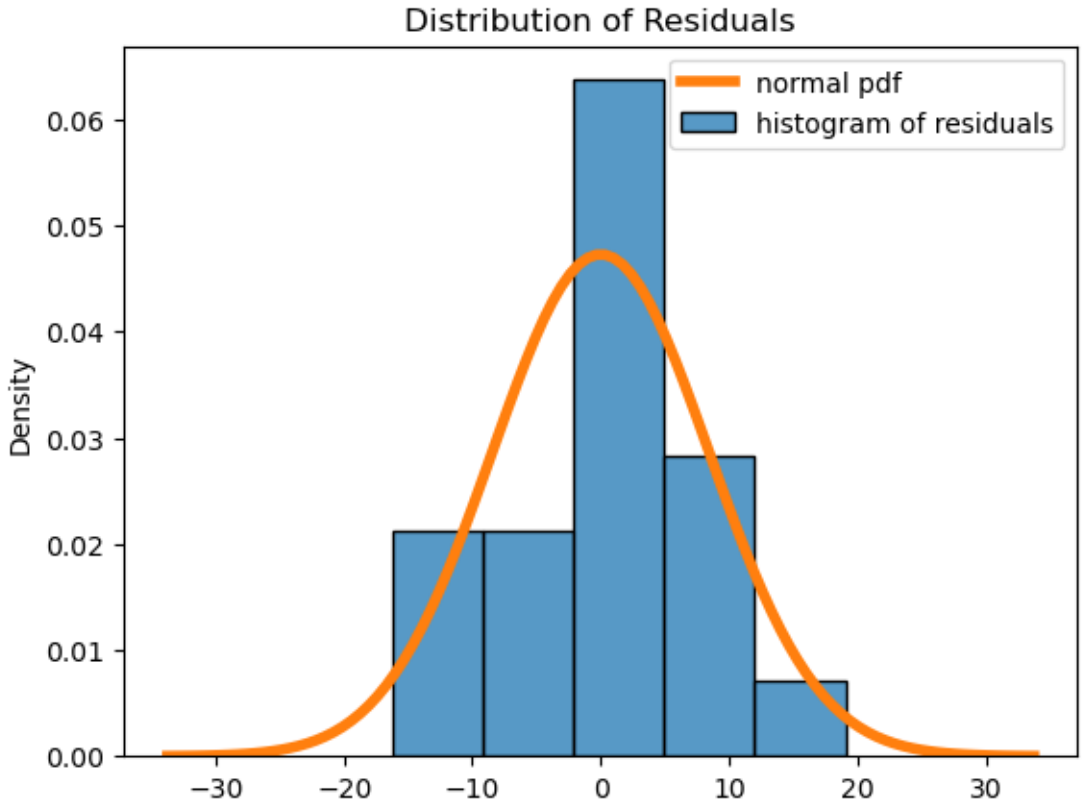
4. What is the value of r^2? Does the model seem to be a good fit? Explain your reasoning.

r^2 is about 0.773, indicating that about 77% of the variation in points is explained by the number of goals a team scores. This is a good fit.

5. Provide diagnostic plots and comment on the results.

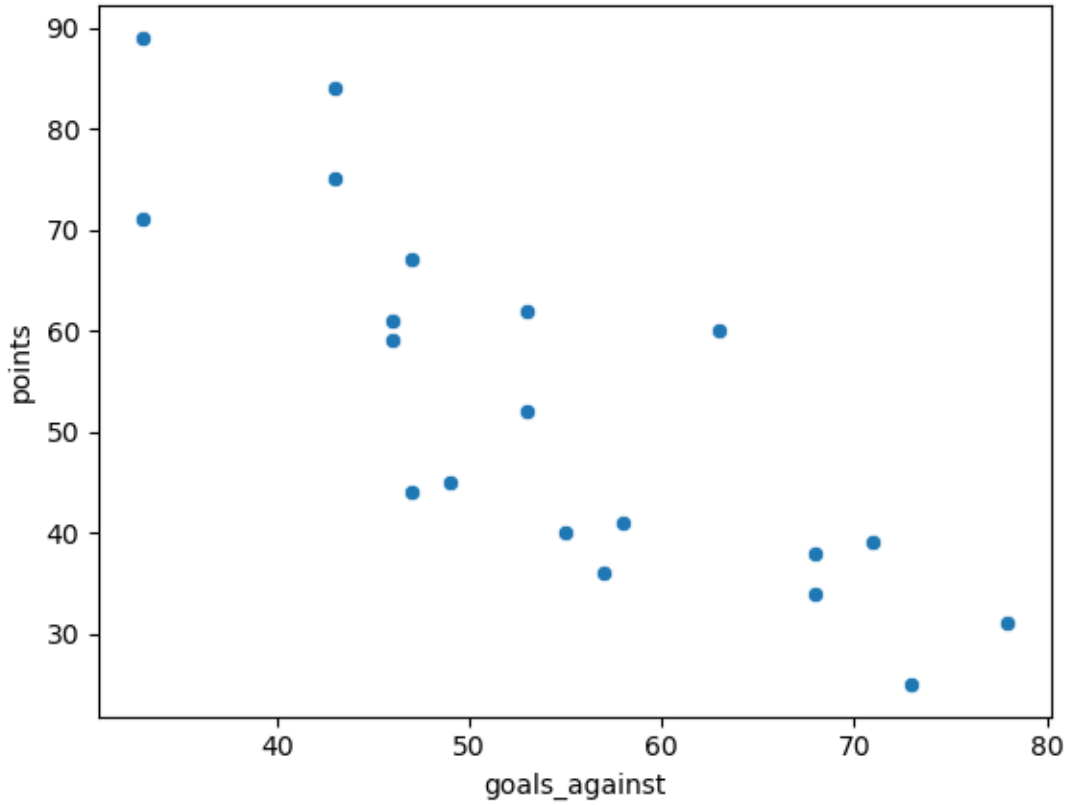
The residuals have no pattern and appear to be roughly mound-shaped. So the assumptions seem valid.


Fitted Values vs. Residuals
Residuals
Fitted Values
20
15
10
5
0
−5
−10
−15
30
40
50
60
70
80


Distribution of Residuals
normal pdf
histogram of residuals
Density
0.06
0.05
0.04
0.03
0.02
0.01
0.00
−30
−20
−10
0
10
20
30

6. Repeat this analysis with another variable or variables. Do your new models seem to be a better or worse fit? Why or why not?

I repeated this using `goals_against`. The fit is slightly worse, with an r^2 of 0.685. Each additional goal against is worth on average 1.17 fewer points in the standings. The diagnostics show that the assumptions are valid. There aren't quite enough points to evaluate the normality of the residuals, but depending on the number of bins in the histogram the plot looks more or less mound-shaped.

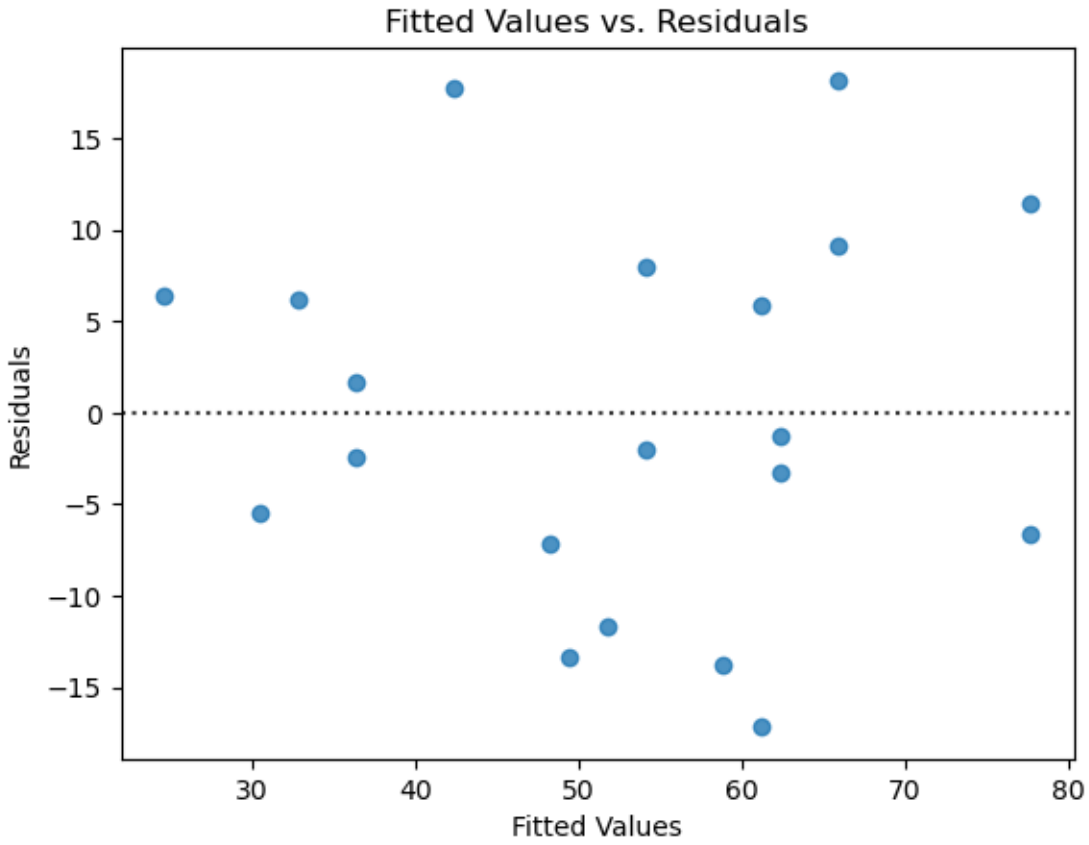


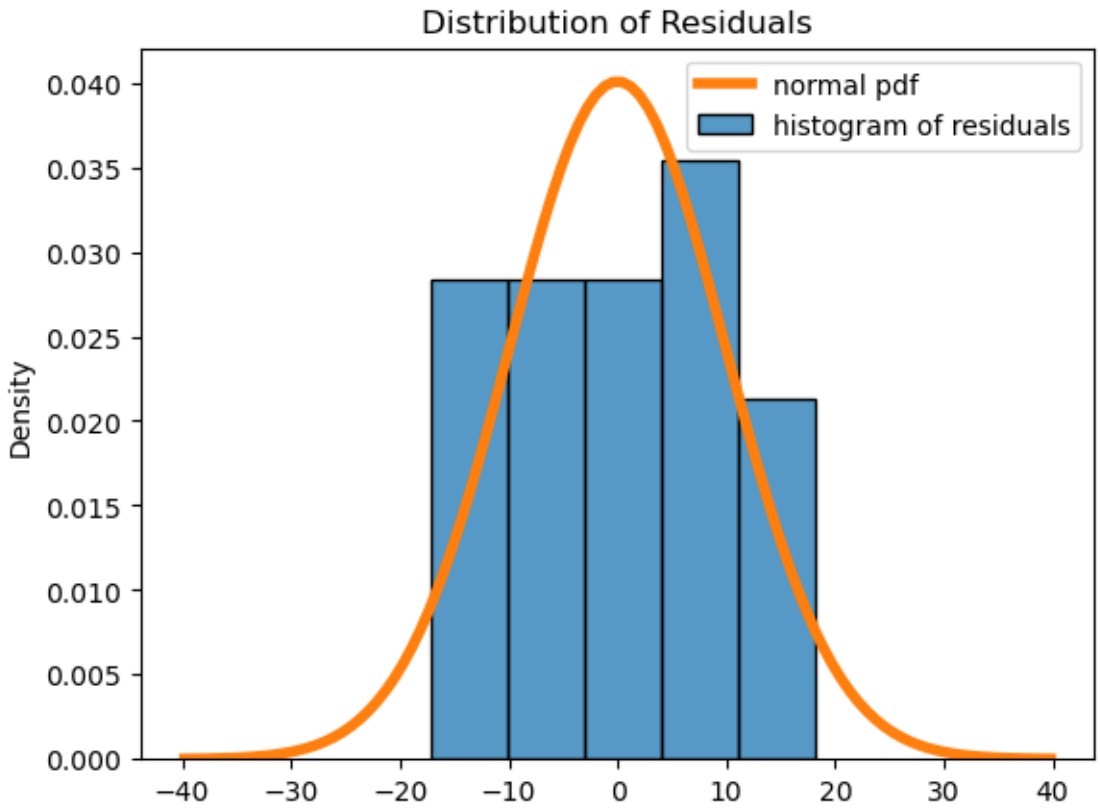

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 points   R-squared:                       0.685
Model:                            OLS   Adj. R-squared:                  0.667
Method:                 Least Squares   F-statistic:                     39.09
Date:                Thu, 20 Jul 2023   Prob (F-statistic):           6.75e-06
Time:                        14:27:50   Log-Likelihood:                -74.341
No. Observations:                  20   AIC:                             152.7
Df Residuals:                      18   BIC:                             154.7
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        116.5044     10.479     11.117      0.000      94.488     138.521
goals_against    -1.1781      0.188     -6.252      0.000      -1.574      -0.782
==============================================================================
Omnibus:                        0.749   Durbin-Watson:                   1.747
Prob(Omnibus):                  0.687   Jarque-Bera (JB):                0.689
Skew:                           0.125   Prob(JB):                        0.709
```

(continues on next page)

(continued from previous page)

```
Kurtosis:                       2.126   Cond. No.                         248.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly␣
↪specified.
```





6. Do you think `attendance` has any effect on a team's performance? Use some of the tools from this section to answer, and explain what you find. How might you use this example to help understand the common phrase "correlation is not causation"?

I would think that attendance has no effect on the results, or perhaps a slightly positive effect. This is shown in the model results, which find a weak r^2 of 0.326, with each additional 1000 people in average attendance worth on average $1000 * .0006 = .6$ points in the standings.

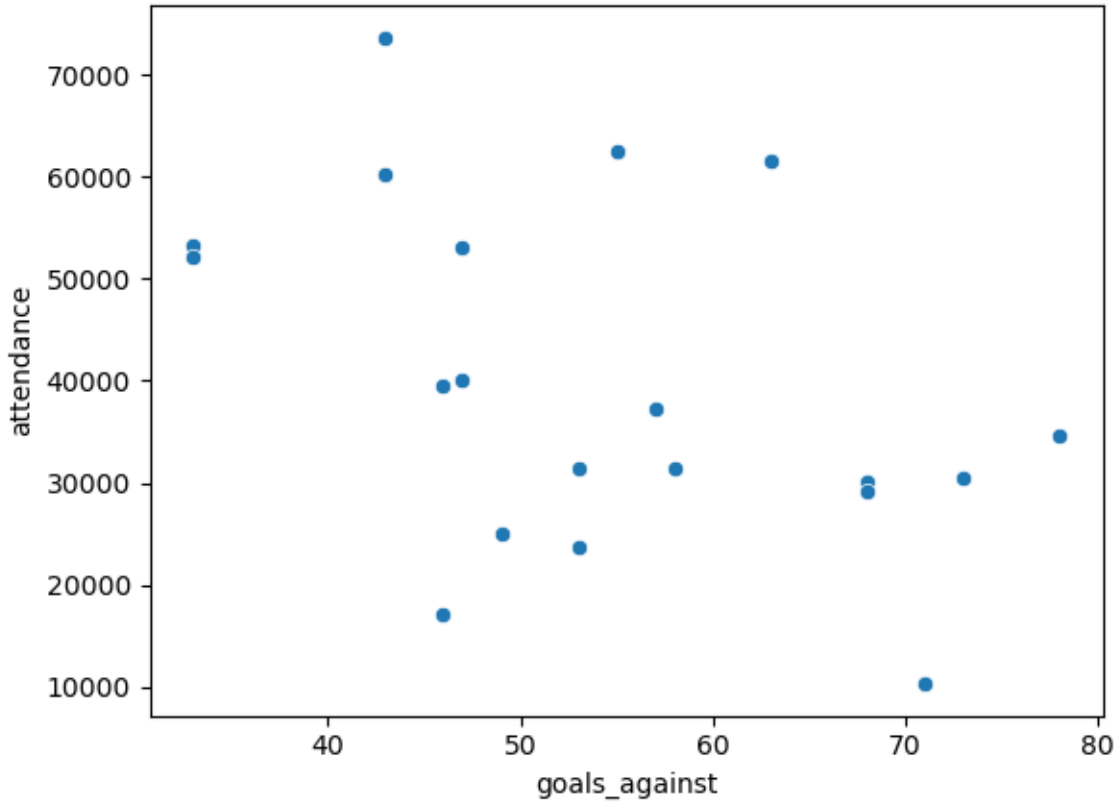
These two variables are correlated, but it is not clear whether one causes the other! Do more people show up to watch good teams, or does more fans cheering help the team win more often? There are lots of things at play here, and no direct causation can be drawn here.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 points   R-squared:                       0.326
Model:                            OLS   Adj. R-squared:                  0.288
Method:                 Least Squares   F-statistic:                     8.697
Date:                Thu, 20 Jul 2023   Prob (F-statistic):            0.00858
Time:                        14:28:26   Log-Likelihood:                -81.941
No. Observations:                  20   AIC:                             167.9
Df Residuals:                      18   BIC:                             169.9
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         28.0933      9.006      3.119      0.006       9.172      47.015
attendance     0.0006      0.000      2.949      0.009       0.000       0.001
==============================================================================
```

(continues on next page)

(continued from previous page)

```
Omnibus:                        0.605   Durbin-Watson:                   1.408
Prob(Omnibus):                  0.739   Jarque-Bera (JB):                0.611
Skew:                           0.010   Prob(JB):                        0.737
Kurtosis:                       2.144   Cond. No.                     1.13e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly␣
↪specified.
[2] The condition number is large, 1.13e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```



Exercise 2

1. How many different levels (i.e., possible categories) are in the `species` variable? How does the software handle the `species` variable?

The species variable has three levels, so the software creates two additional dummy variables to handle this.

```
   flipper_length_mm  sex_male  species_Chinstrap  species_Gentoo
0              181.0         1                  0               0
1              186.0         0                  0               0
2              195.0         0                  0               0
3                NaN         0                  0               0
4              193.0         0                  0               0
```

2. Are all the variables providing useful information in your model? How can you tell?

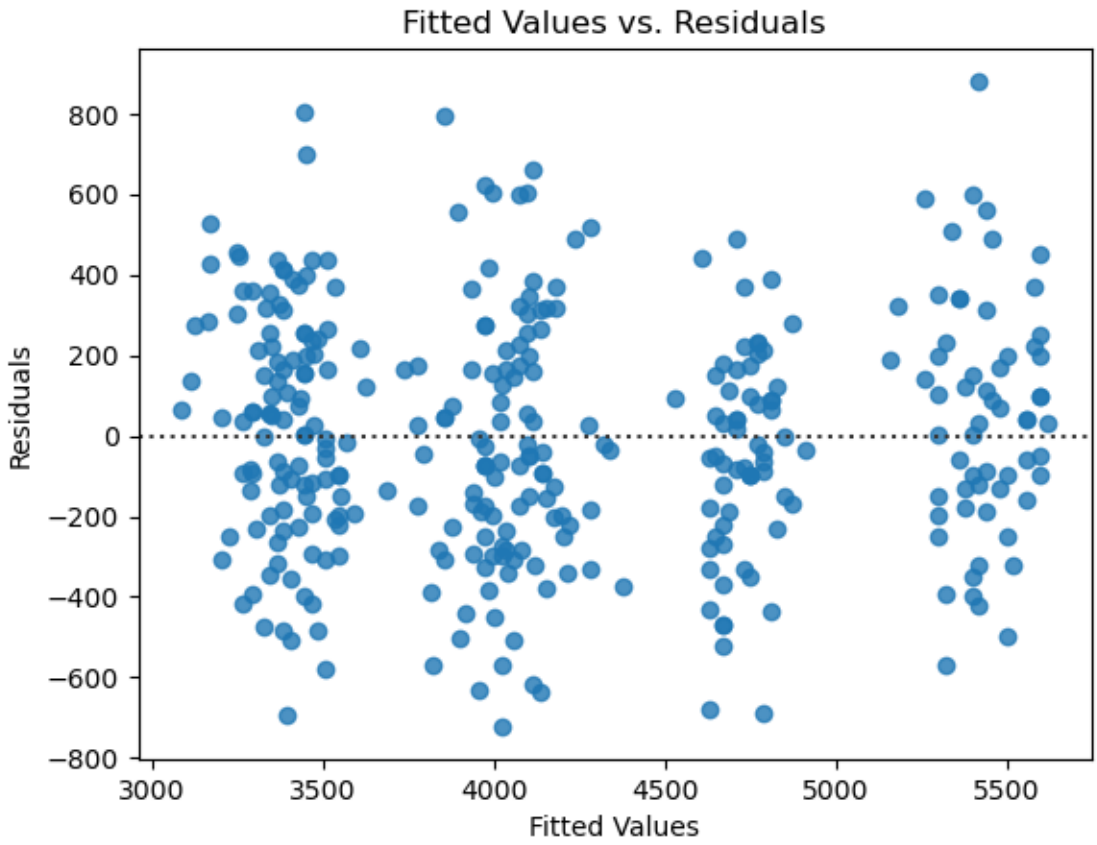
It looks like all the variables are providing useful information based on the P-values. However, the intercept doesn't appear to be useful.

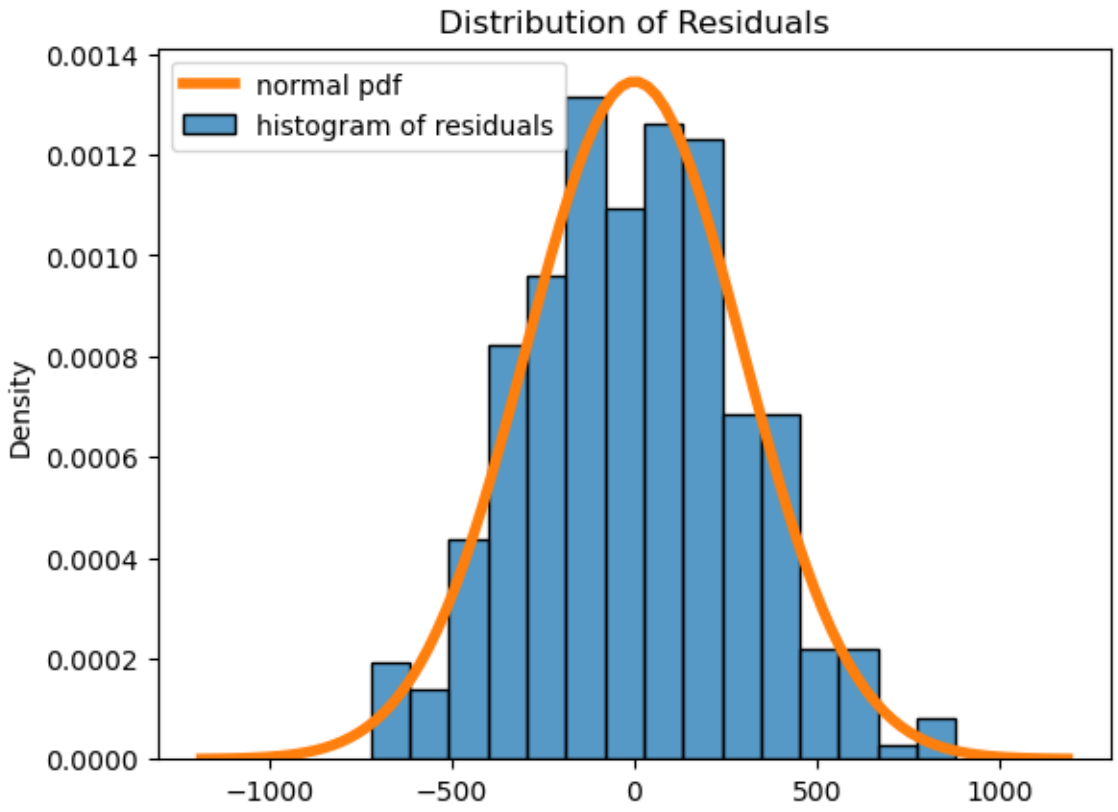
```
                            OLS Regression Results
==============================================================================
Dep. Variable:            body_mass_g   R-squared:                       0.863
Model:                            OLS   Adj. R-squared:                  0.861
Method:                 Least Squares   F-statistic:                     530.0
Date:                Thu, 20 Jul 2023   Prob (F-statistic):          6.14e-144
Time:                        14:27:51   Log-Likelihood:                -2432.0
No. Observations:                 342   AIC:                             4874.
Df Residuals:                     337   BIC:                             4893.
Df Model:                           4
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.
↪975]
------------------------------------------------------------------------------------------------
↪--
const                         -357.1853    533.378     -0.670      0.504   -1406.355     691.
↪985
flipper_length_mm               20.0139      2.853      7.014      0.000      14.401      25.
↪627
sex_male                       529.8248     37.745     14.037      0.000     455.579     604.
↪071
species_Chinstrap              -93.8251     46.625     -2.012      0.045    -185.537      -2.
↪113
species_Gentoo                 823.6912     85.544      9.629      0.000     655.423     991.
↪959
==============================================================================
Omnibus:                        1.202   Durbin-Watson:                   2.105
Prob(Omnibus):                  0.548   Jarque-Bera (JB):                1.299
Skew:                           0.113   Prob(JB):                        0.522
Kurtosis:                       2.800   Cond. No.                     6.72e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly␣
↪specified.
[2] The condition number is large, 6.72e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

3. Describe the typical characteristics of a very large or very small penguin.

A penguin with large mass would be a male Gentoo with large flipper length. A small penguin would be a female Chinstrap with small flipper length.

4. Provide diagnostics for your model and comment on the validity of the regression assumptions.





The diagnostics look great!

6.7 JNB Lab: The Framingham Heart Study

The Framingham Heart Study is an ongoing study of cardiovascular health in the United States. The initial study followed over 5,000 volunteers from Framingham, Massachusetts, USA over the course of several decades, and it still continues today. The study led to important findings in many areas, including a link between cholesterol and heart disease. (These exercises are inspired by an assignment from UC Berkeley's Data 8.)

We load in the data below from the file `framingham.csv` (click here to download a copy). There are a number of different interesting variables to explore, but we will focus on exploring total cholesterol levels (`TOTCHOL`) versus the occurrence of heart disease (`ANYCHD`). The variable `ANYCHD` takes the value `0` if the patient does not have heart disease and the value `1` if they do.

```
framingham = pd.read_csv("framingham.csv")
framingham.head()
```

	AGE	SYSBP	DIABP	TOTCHOL	CURSMOKE	DIABETES	GLUCOSE	DEATH	ANYCHD
0	39	106.0	70.0	195.0	0	0	77.0	0	1
1	46	121.0	81.0	250.0	0	0	76.0	0	0
2	48	127.5	80.0	245.0	1	0	70.0	0	0
3	61	150.0	95.0	225.0	1	0	103.0	1	0
4	46	130.0	84.0	285.0	1	0	85.0	0	0

6.7.1 Part 1: Explore the data

As always, it's important to take a look at your data before you dive into any kind of inference. We first note that we have both categorical and numerical variables, which changes what types of visualization we might be interested in. First, we'll look at the size of the data frame, which shows that we have 3842 observations of 9 variables.

```
framingham.shape
```

```
(3842, 9)
```

Since we are interested in how cholesterol connects to occurrence of heart disease, we next explore those variables. Like the penguin example in the text, we will separate our data into our two samples. Then we'll calculate the mean cholesterol rating for each group.

```
chd = framingham[framingham['ANYCHD'] == 1]
nochd = framingham[framingham['ANYCHD'] == 0]

print(f"The mean total cholesterol for those with an occurrence of CHD is {chd[
 ↪'TOTCHOL'].mean():.3f}.")
print(f"The mean total cholesterol for those without occurrence of CHD is {nochd[
 ↪'TOTCHOL'].mean():.3f}.")
```

```
The mean total cholesterol for those with an occurrence of CHD is 249.482.
The mean total cholesterol for those without occurrence of CHD is 232.846.
```

The means are different, but it's not clear if they are a lot different, or just a little bit. That depends on the sample sizes and the distribution of cholesterol levels!

1. Write a line or two of code to figure out how many people in the study have an occurrence of CHD and how many do not.
2. Make a histogram of the cholesterol levels for both samples. Describe the distributions' centers, shapes, and compare the two.

6.7.2 Part 2: Two Sample T-tests

We want to determine if there is a true difference in average cholesterol levels between people with heart disease and those without. This is true in our sample, but we need to see if the evidence is enough to make a claim about the population. We are testing the hypotheses:

$H_0 : \mu_1 = \mu_2$ $H_1 : \mu_1 \neq \mu_2$

where μ_1 and μ_2 represent the average total cholesterol levels of the CHD and No CHD populations, respectively. But first we need to know if using a T-test is valid!

1. Describe the assumptions of the T-test and comment on if they are valid for this example. Your work in Part 1 should be enough.

Once we know that the test is appropriate for our problem, we want to perform the test. Recall that to do this we want to set up a `CompareMeans` object from our two samples. Just like in the chapter, we do that in the following way.

```
sample1 = chd['TOTCHOL']
sample2 = nochd['TOTCHOL']

# create a CompareMeans object from the two samples
cm = sm.stats.weightstats.CompareMeans.from_data(sample1, sample2)
```

Now we are ready to perform the test!

2. Compute the test statistic and P-value using `cm.ttest_ind`.
3. Write your conclusion in a complete sentence.
4. Give a confidence interval for the difference in average total cholesterol.

6.8 JNB Lab Solutions

```
framingham = pd.read_csv("framingham.csv")
```

```
   AGE  SYSBP  DIABP  TOTCHOL  CURSMOKE  DIABETES  GLUCOSE  DEATH  ANYCHD
0   39  106.0   70.0    195.0         0         0     77.0      0       1
1   46  121.0   81.0    250.0         0         0     76.0      0       0
2   48  127.5   80.0    245.0         1         0     70.0      0       0
3   61  150.0   95.0    225.0         1         0    103.0      1       0
4   46  130.0   84.0    285.0         1         0     85.0      0       0
```

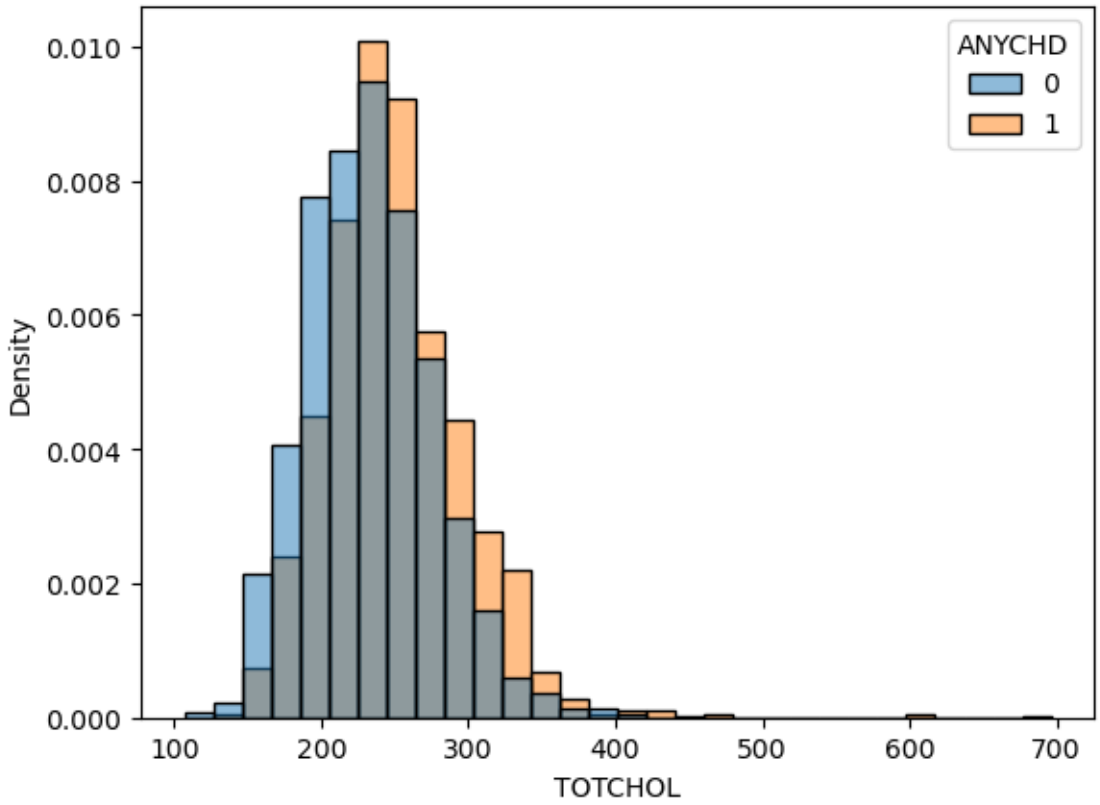
6.8.1 Part 1: Explore the data

1. Write a line or two of code to figure out how many people in the study have an occurrence of CHD and how many do not.

There are 954 people in the study with CHD and 2888 without CHD.

```
954 2888
```

2. Make a histogram of the cholesterol levels for both samples. Describe the distributions' centers, shapes, and compare the two.



6.8.2 Part 2: Two Sample T-tests

1. Describe the assumptions of the T-test and comment on if they are valid for this example. Your work in Part 1 should be enough.

The t-test is valid since we have a large sample size in both samples ($n_1 = 945$ and $n_2 = 2888$), and both sample distributions are relatively mound shaped.

2. Compute the test statistic and P-value using `cm.ttest_ind`.

```
Results of T-test: test statistic is 9.766 with 1545.774 degrees of freedom.
P-value is 0.000.
```

3. Write your conclusion in a complete sentence.

Solution: Our test yields a test statistic of 9.766 on 1545.774 degrees of freedom, with a p-value of 0. Thus, we reject the null hypothesis that the average total cholesterol levels are the same in both populations.

4. Give a confidence interval for the difference in average total cholesterol.

Solution: A 95% confidence interval for the difference is (13.295, 19.977), meaning that we are 95% confident that average cholesterol levels are (roughly) 13 to 20 points higher in patients with CHD.

```
95% confidence interval is (13.295, 19.977).
```

CHAPTER 7

APPLIED CALCULUS FOR DAILY LIFE - Wheaton College Team

Note

The material in this chapter was developed for use in an introductory level applied calculus course by a Wheaton College team comprised of Dr. Paul Isihara (lead professor) with Gavin McClowry, Samuel Carlson and Ziling Zhong.

7.1 Introduction

This chapter takes mathematical principles explored in calculus (graphing lines and linear systems, optimization, related rates, area between two curves, integrals, exponential growth and decay, etc.) and shows their real life applications:

- Linear Systems and the Two-commodity Model: Explains how linear systems can be useful in budgeting and shifts in basic economic systems
- Marginal and Average Cost: Uses derivatives to introduce the concept of marginal functions
- Optimization and Object Design: Uses optimization to help design architecture
- Optimization and Cobbs-Douglas Production: Finds minimums and maximums in production output using optimization
- Related Rates and Volumes: Shows how to analyze changing volumes using related rates
- Related Rates with Trig Functions and Football: Shows how related rates can be useful in analyzing sports plays
- Probability Distributions and Drive Thrus: Uses probability techniques to simulate drive thru orders
- Normal Distribution and Process Control: Introduces the concept of normal distribution and its relevance to process control in manufacturing
- Partial Derivatives and OLS Regression: Shows an application of partial derivatives in data analysis (Ordinary Least Squares)
- Area Between Curves and the Gini Index: Uses the calculus technique of finding the area between two curves to represent inequity

- Integral Test and Income Streams: Introduces an application of the integral test and convergence/divergence to income and long-range investment
- Ordinary Differential Equations and Exponential Growth/Decay: Introduces the use of exponential growth and decay models in pollution and economic modeling

Preview

Calculus as we know it today would not have been developed without the Cartesian coordinate system and analytic geometry. Isaac Newton (1642-1727) attributed his success to "standing on the shoulders of giants" such as Descartes (1596-1650). In this section we explore how a concept as simple as graphing straight lines in a Cartesian plane is helpful in analyzing a budgeting problem called the "Two Commodity Model" [Vali 2014]. The simple idea that changing the coefficients in a linear relation will affect its graph (a simple example of "sensitivity analysis") becomes important when considering changes in unit prices and also in total budget amounts.

Reference: Vali, Shapor, 2014. Principles of Mathematical Economics. Paris: Atlantis Press.

7.2 Linear Systems and the Two-commodity Model

7.2.1 Two-Commodity Model

Suppose a household consumer is budgeting for two types of goods, necessities and luxuries. Let X denote the units of necessities and Y denote the units of luxuries. We let c_X denote the cost per unit of necessity and c_Y the cost per unit of luxury. If the total monthly budget is B, then the budget equation is

$$c_X X + c_Y Y = B.$$

Let $r = \frac{Y}{X}$ be the ratio of units of luxury to units of necessities. Then the **income consumption path** is the line

$$\frac{Y}{X} = r \Rightarrow Y = rX.$$

The **equilibrium bundle** is defined to be the point of intersection between the budget equation and the income consumption path.

Example

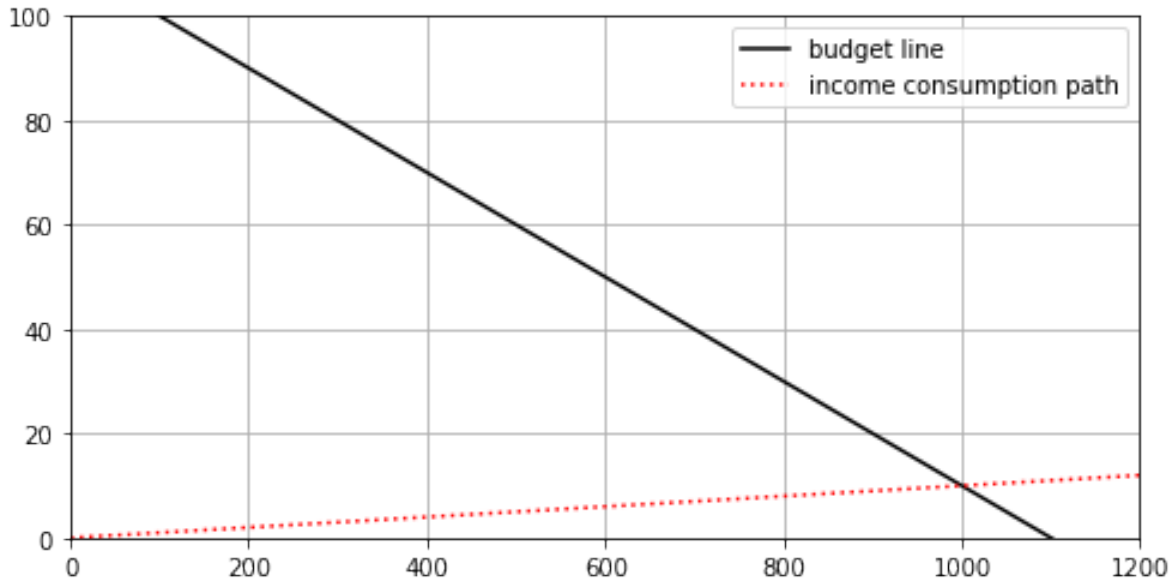
Find the equilibrium bundle if $c_X = 10$, $c_Y = 100$, $B = 11000$, and $r = \frac{1}{100}$.

Solution. In this case, the equilibrium bundle is the solution to the linear system

$$10X + 100Y = 11000 \qquad Y = \frac{1}{100}X.$$

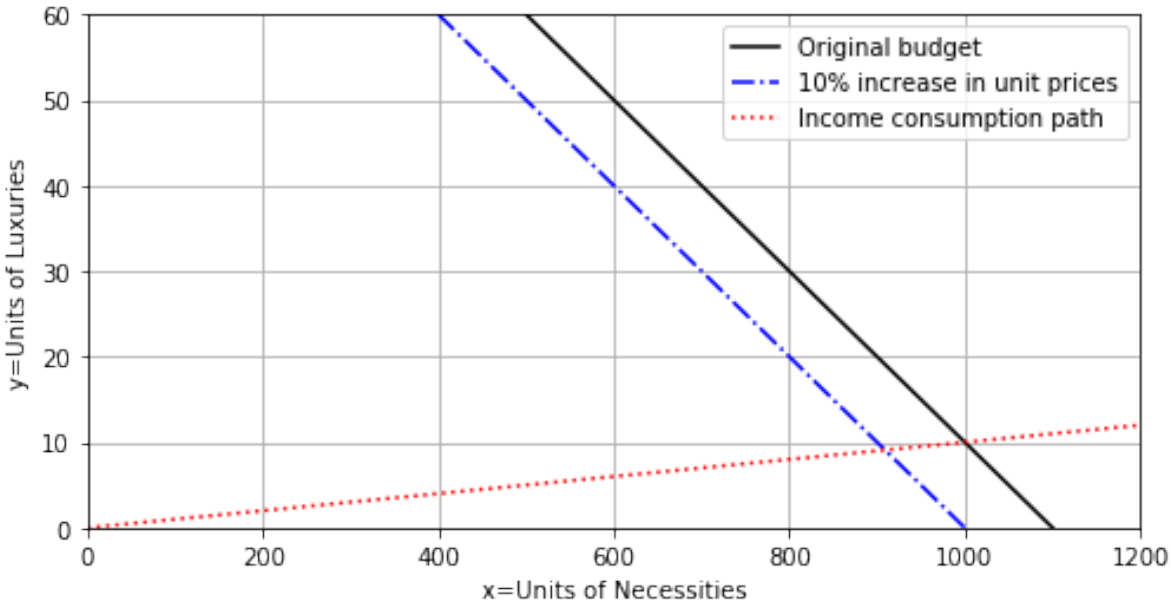
This system is easily solved to yield the equilibrium bundle (1000,10).

Here we plot the original budget constraint line and income consumption path.



7.2.2 Sensitivity Analysis

A *sensitivity analysis* examines how a model is affected when one or more of its parameters are changed. For example, suppose due to inflation the cost per unit of necessity ($c_X = 10$) and luxury $c_Y = 100$ both increase by 10%. Then the new budget equation is $11X + 110Y = 11000$ and the new equilibrium bundle is approximately (909,9.1). This makes sense since higher unit prices mean less units of each type of item can be purchased.



Exercises

The following questions will compare the effect of increasing the budget vs. decreasing unit prices and also decreasing the budget vs. increasing the unit prices. Suppose that the monthly budget is $4,000 and the unit price for necessities is $30 and for luxuries it is $100.

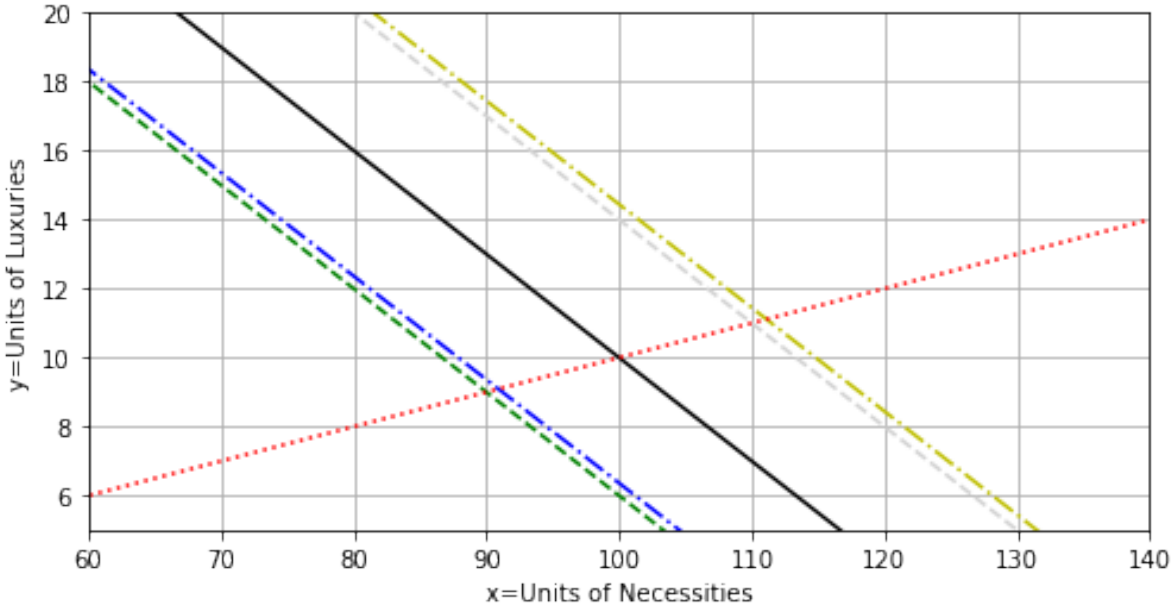
1) Determine the equilibrium bundle if the ratio of necessities to luxuries is 10 to 1.

2) Which has the greater impact on the bundle: A 10% decline in budget or a 10% increase in unit prices?

3) Which has the greater impact on the bundle: A 10% increase in budget or a 10% decrease in unit prices?

4) Explain your answers to the above using the figure below.

Reference Vali, Shapor, 2014. Principles of Mathematical Economics. Paris: Atlantis Press.



7.3 Marginal and Average Cost

7.3.1 Derivative Rules

Given a function $y = f(x)$, the derivative $f'(x)$ evaluated at $x = a$ gives the slope of the tangent to the graph of $y = f(x)$ at the point $(a, f(a))$.

There are rules to quickly finding the derivative of polynomials and functions with powers.

If $f(x) = mx + b$, then $f'(x) = m$.

If $f(x) = kx^n$, then $f'(x) = knx^{n-1}$.

If $f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$, then $f'(x) = a_1 + 2a_2x + 3a_3x^2$.

(Note that the derivative of the constant term by itself is 0. The same pattern continues for higher powers).

EXAMPLE: If $f(x) = 2x^3 - 5x^2 + 2x + 10$, then $f'(x) = 6x^2 - 10x + 2$.

EXAMPLE: If $f(x) = \frac{2}{\sqrt{x}} = 2x^{-.5}$, then $f'(x) = 2(-.5)x^{-1.5} = \frac{-1}{x^{1.5}}$

7.3.2 Marginal Functions

The term ``marginal" is used to mean derivative.

EXAMPLE: Let TC be total cost and Q be the number of units produced. If $TC(Q) = 50 + 10Q^{3/2}$, then the marginal cost is $MC(Q) = TC'(Q) = 10(3/2)Q^{1/2} = 15\sqrt{Q}$

Note: Since the derivative of a function $f(x)$ is given by

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h},$$

if we choose $h = 1$, then we get an approximation

$$f'(x) \approx f(x+1) - f(x).$$

Thus, marginal functions can be used to approximate the difference between the value of the original function at $x + 1$ and the value of the original function at x.

For the total cost function, the derivative $MC(Q) = TC'(Q)$ approximates the difference in total cost producing $Q + 1$ units and producing Q units. In other words, the marginal total cost approximates the the total cost to produce 1 more unit if the current production level is Q.

7.3.3 Fixed Cost

The term ``fixed cost" means the cost incurred when $Q = 0$ units are produced.

EXAMPLE: Let $TC(Q) = 50 + 10Q^{3/2}$. Then the fixed cost is $TC(0) = 50$.

7.3.4 Demand Function

A demand function $P(Q)$ gives the price P per unit when selling Q units. The total revenue TR generated by selling Q units is therefore $TR = Q \cdot P(Q)$.

Example 1. Find the total revenue TR and marginal revenue MR if the demand function is $P = 100 - 5\sqrt{Q}$.

Answer: $TR = Q(100 - 5\sqrt{Q}) = 100Q - 5Q\sqrt{Q} = 100Q - 5Q(Q^{\cdot 5}) = 100Q - 5Q^{1.5}$.

$MR = TR' = 100 - 5(1.5)Q^{\cdot 5} = 100 - 7.5Q^{\cdot 5}$.

Example 2. Algebra might be needed to find the demand function $P(Q)$.

If $Q = \frac{10}{P^2}$, then $P^2 = \frac{10}{Q} \Rightarrow P = \frac{\sqrt{10}}{\sqrt{Q}}$. The total revenue is therefore $TR = Q \cdot P = Q(\frac{\sqrt{10}}{\sqrt{Q}}) = \sqrt{Q}\sqrt{10} = \sqrt{10}Q^{1/2}$. It follows that the marginal revenue is $MR = \frac{1}{2}\sqrt{10}Q^{-1/2} = \frac{\sqrt{10}}{2\sqrt{Q}}$.

7.3.5 Average Cost

The average cost equals the total cost divided by number of units produced.

EXAMPLE. If the total cost is $TC = 10Q^2 + 5Q + 10$, then the average cost is $AC = \frac{TQ}{Q} = 10Q + 5 + \frac{10}{Q}$

7.3.6 Graphical Intersections

To find a point where the graph of $f(x)$ intersects the graph of $g(x)$, solve the equation $f(x) = g(x)$. (For our applications we require that $x \geq 0$)

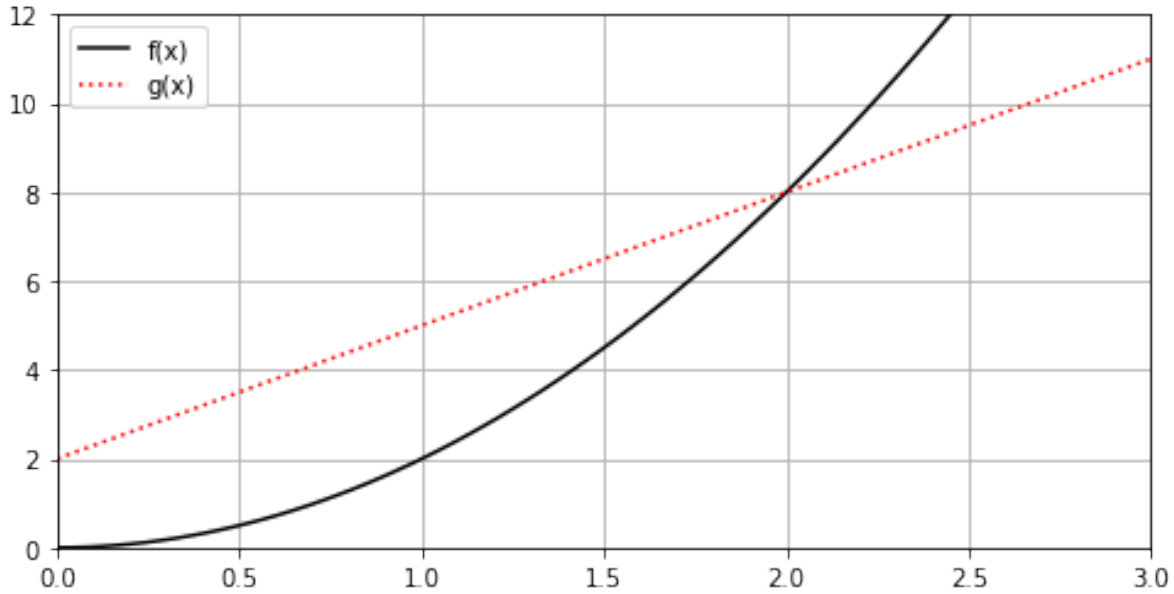
EXAMPLE: Find where $f(x) = 2x^2$ and $g(x) = 3x + 2$ intersect.

Solution. We must solve $2x^2 = 3x + 2$. That is $2x^2 - 3x - 2 = 0$. To solve $ax^2 + bx + c = 0$, we can use the quadratic formula $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$. In our case,

$$x = \frac{3 \pm \sqrt{9 + 16}}{4} = \frac{3 \pm 5}{4} = 2$$

(We do not use the other root at $x = -1/2$ since we require $x \geq 0$).

The y-coordinate of the intersection point with $x = 2$ is $y = f(2) = 2(2^2) = 8$. Note that $g(2) = 3(2) + 2 = 8$ so in fact, the graph of both $f(x)$ and $g(x)$ includes the point $(2, 8)$



7.3.7 Exercises

Exercises

1) Consider the total cost function

$$TC = .5Q^3 - 3Q^2 + 12Q + 50.$$

Find and graph the average total cost ATC and the marginal cost MC. Use the graph to determine where $ATC = MQ$. Then show that ATC has a minimum value at the point of intersection. (That is, show that $ATC'(Q) = 0$ at the point of intersection).

2) Repeat problem 1 using the total cost function

$$TC(Q) = -10Q^3 + 128Q^2 + 3Q.$$

(In this case, the average total cost has a maximum rather than minimum value at the point of intersection.)

3) The quotient rule says that

$$\frac{d}{dx}\frac{f(x)}{g(x)} = \frac{g(x)f'(x) - f(x)g'(x)}{(g(x))^2}.$$

Use the quotient rule to prove that the average total cost must have a max or min at the point where it equals the marginal cost.

7.4 Optimization and Object Design

Intro

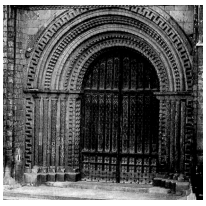
Optimization is the problem of finding a maximum or minimum value of an objective function, and is one of the main applications of differential calculus. In this section we will consider optimal design of several objects encountered in every day life.

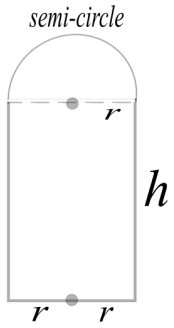
7.4.1 Example 1: Cathedral Door

Example 1: Cathedral Door

Suppose we wish to design a cathedral door with a rectangular base and semi-circular top. How can we maximize the area A of the door for a given perimeter P?

This can be solved using calculus as outlined below.





1) First note that the perimeter P is kept constant throughout this problem. The radius r and height h must satisfy the equation

$$P = 2r + 2h + \pi r.$$

This implies that $P = (2+\pi)r + 2h$ or $2h = P - (2+\pi)r$. This tells us the height h of the door is related to the

constant perimeter P and radius r as follows:

$$h = \frac{P}{2} - (1 + \frac{\pi}{2})r.$$

2) Next note that the total area A of the door is the sum of the rectangular base with area $2rh$ and semi-circular top with area $\frac{1}{2}\pi r^2$:

$$A = 2rh + \frac{1}{2}\pi r^2.$$

We can replace the height h with the expression we obtained in step 1):

$$A = 2r[\frac{P}{2} - (1 + \frac{\pi}{2})r] + \frac{1}{2}\pi r^2.$$

We simplify the right side of the last equation as follows:

$$A = Pr - r^2(2 + \pi) + \frac{1}{2}\pi r^2$$

or

$$A = Pr - r^2(2 + \frac{\pi}{2}).$$

3) Keeping in mind that P is a constant, we compute the derivative $\frac{dA}{dr}$ and set it equal to zero:

$$\frac{dA}{dr} = P - 2r(2 + \frac{\pi}{2}) = 0.$$

This gives us the optimal radius r of the door in terms of the fixed perimeter P:

$$P = 2r(2 + \frac{\pi}{2}) = r(4 + \pi) \quad \Rightarrow \quad r = \frac{P}{4 + \pi}.$$

4) The corresponding value for the height h of the door is

$$h = \frac{P}{2} - (1 + \frac{\pi}{2})r = \frac{P}{2} - (1 + \frac{\pi}{2})\frac{P}{4 + \pi} = \frac{P[4 + \pi - 2(1 + \frac{\pi}{2})]}{2(4 + \pi)} = \frac{P}{4 + \pi}.$$

(Note that the optimal radius and height are equal to each other: the door is low and wide.)

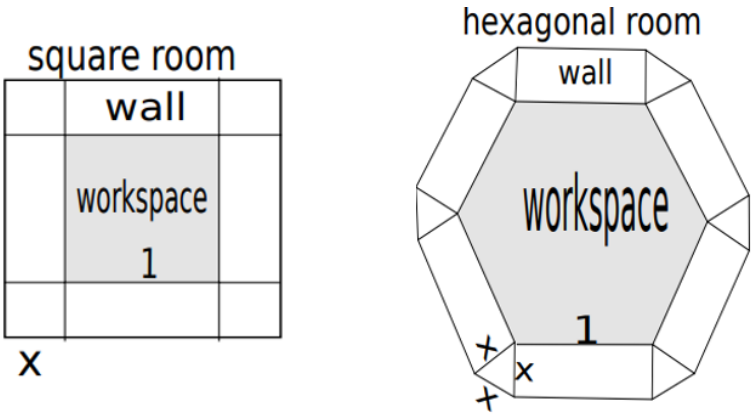
5) Finally, note that the second derivative is negative, guaranteeing that we have maximized rather than minimized the area:

$$A''(r) = -2(2 + \frac{\pi}{2}) < 0.$$

7.4.2 Example 2: Office Space Design

Example 2: Office Design

Consider the problem of designing an office space. We will consider two designs, and decide which is better on the basis of lower cost of wall materials per unit area of enclosed workspace. For simplicity, assume that the cost of wall materials is 1 monetary unit per unit of wall base area.



For the square carrel, the area of the wall is $4x^2 + 4x$ and the enclosed workspace is 1 square unit, so the cost of materials per square foot of workspace is

$$f(x) = 4x^2 + 4x.$$

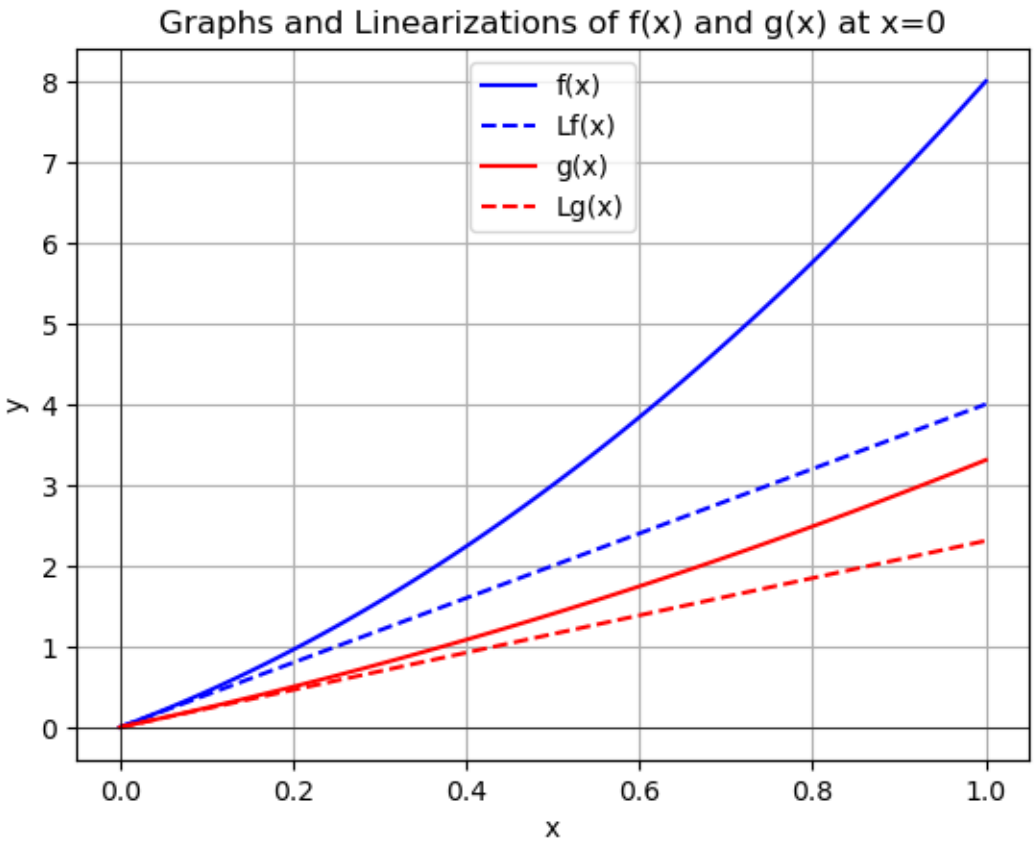
For the hexagonal carrel, the area of the wall is $3x^2\frac{\sqrt{3}}{2} + 6x$ and the enclosed workspace has area $\frac{3\sqrt{3}}{2}$. Thus, the cost of materials per square foot of workspace is

$$g(x) = \frac{3x^2\frac{\sqrt{3}}{2} + 6x}{\frac{3\sqrt{3}}{2}} = x^2 + \frac{4x}{\sqrt{3}}.$$

Note that both functions start at the origin, and for any wall thickness $x > 0$, we must have $f(x) > g(x)$. We can argue this analytically since we first compute $f'(x) = 8x + 4$, $f''(x) = 8$ and $g'(x) = 2x + \frac{4}{\sqrt{3}}$ and $g''(x) = 2$. Note that $f'(0) = 4$ and $f''(x) = 8$ for all x. On the other hand, $g'(0) = 2 + \frac{4}{\sqrt{3}} < 4$ and $g''(x) = 2$ for all x. Thus the graph of $f(x)$ has a steeper tangent than $g(x)$ at $x = 0$. Moreover, the concavity of $f(x)$ is greater than the concavity of $g(x)$ for all $x \geq 0$. This guarantees that $f(x) > g(x)$.

Graphically, we can use code from Section 4 of the Calculus chapter to confirm this relationship.

```
Equation of Lf(x): 4 x
Equation of Lg(x): 2.3094010767585 x
```



7.4.3 Exercises

Exercises

1. Suppose a bulk food container is in the shape of a right-circular cylinder and designed to hold 250 cubic inches. Our goal is to find the dimensions of the can which will minimize the amount of material needed for construction.

a) Make a sketch of the can, using x for the radius of the can and h for the height.

b) Given that the volume V of the can is 250 cubic inches, find a formula for h in terms of x.

c) Find a function $f(x)$ which gives the total area of the can (top, bottom, and side) in terms of x.

d) Compute $f'(x)$.

e) Find x such that $f'(x) = 0$.

f) Find h corresponding to the x in part 5).

g) What is the relationship between the diameter of the can (2x) and the height h?. Describe the shape of the bulk food can.

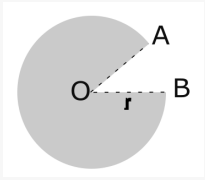
h) Use the second derivative to explain why your answer gives a minimum amount of material used to construct the can.

2) Suppose you would like to design a paper cup using a circular piece of paper of radius r, cutting out a sector from the circle, and joining the two radial edges OA and OB.

a) What is the maximum volume of the cup?

b) Does the second derivative test apply?

c) Make a graph of the volume of the cup as a function of its height h for the case $r = 1$.



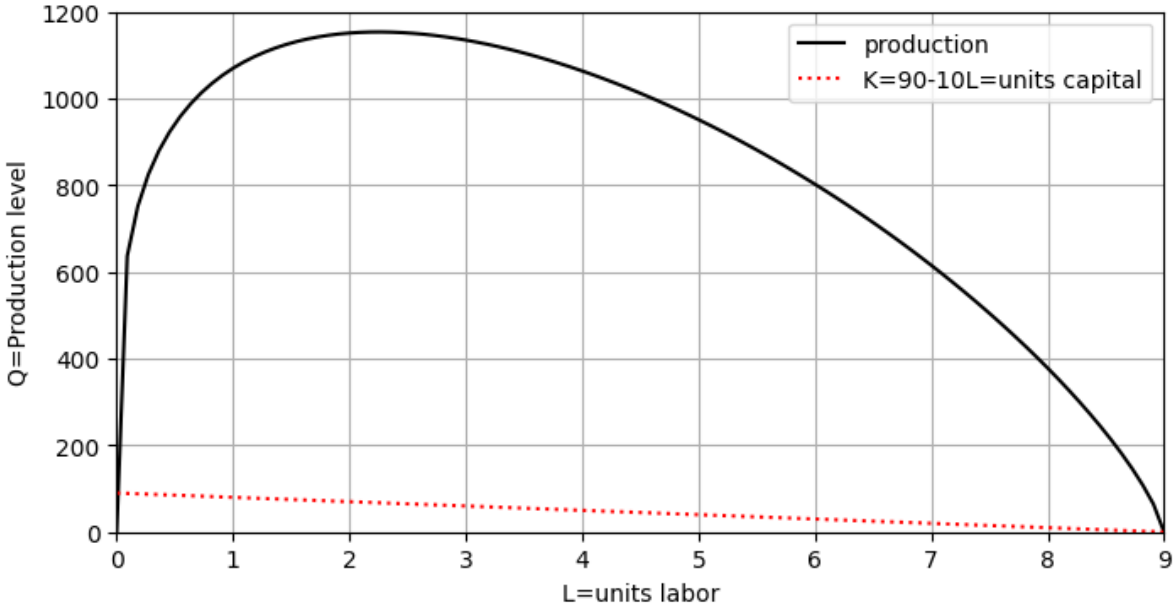
7.5 Optimization and Cobbs-Douglas Production

In this lab we will consider two optimization problems related to a Cobbs-Douglas production function $Q = AL^a K^{1-a}$ where Q is units of production, $L =$ units of labor, $K =$ units of capital and A and α are constants. We also use a total cost function of the form $TC = FC + c_L L + c_K K$ where $FC =$fixed cost, $c_L =$ cost per unit of labor and $c_K =$ cost per unit of capital.

Problem 1: By constraining the total cost, we can obtain a formula for K in terms of L, in which case Q becomes a function of L. A plot of $Q(L)$ shows there is a maximum value. The exact coordinates of the maximum can be obtained by setting $\frac{dQ}{dL}$ equal to zero and solving for L.

Example: Plot $Q = 40L^{1/4}K^{3/4}$ under the total cost constraint $TC = 1000 = 100 + 100L + 10K \Rightarrow K = 90 - 10L$.

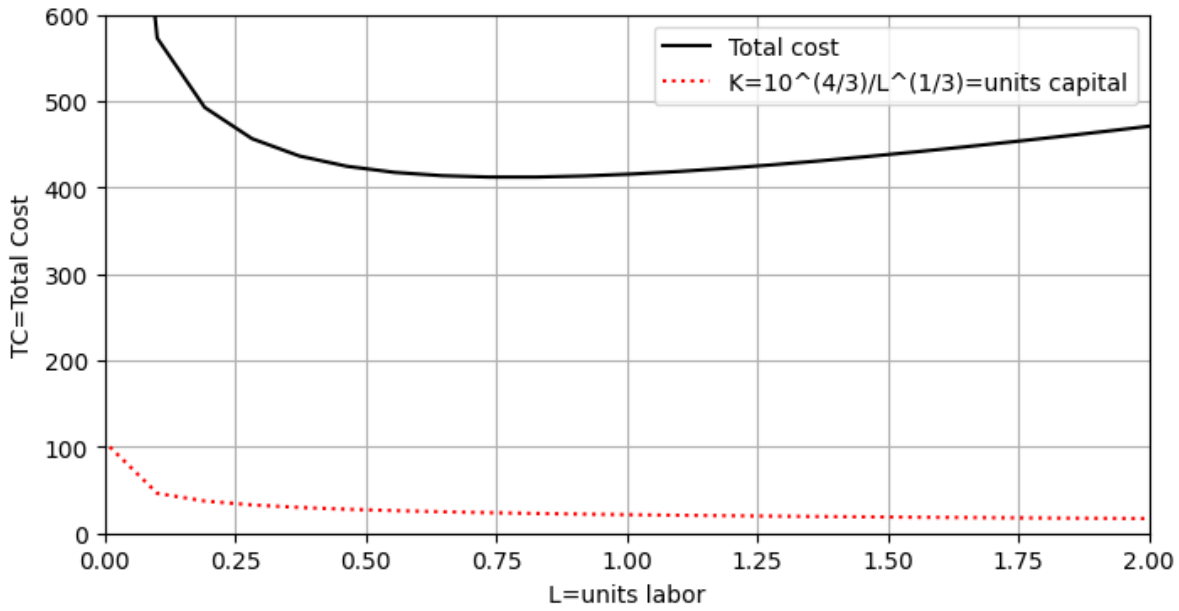
Solution: In this case, $Q = 40L^{\cdot 25}(90 - 10L)^{\cdot 75}$, whose graph created in the next cell has a maximum value.



Problem 2: If we set the production level Q to be a specific value, we can solve for K in terms of L and express the total cost function as a function of L only. A plot of the total cost function $TC(L)$ shows it has a minimum value.

Example: Plot $TC(L, K) = 100 + 100L + 10K$ under the constraint $Q = 40L^{1/4}K^{3/4} = 400 \Rightarrow K = \frac{10^{4/3}}{L^{1/3}}$.

Solution: In this case $TC = 100 + 100L + 10^{7/3}L^{-1/3}$, and the graph created in the next cell shows there is a minimum value.



7.5.1 Exercises

Exercises

1) (Maximize Production given Total Cost) Consider the Cobb-Douglas production function $Q = 2L^{.4}K^{.6}$. Suppose the labor input cost is $c_L = \$10$ per unit and the capital input cost is $c_K = \$20$ per unit. Suppose further that there is a $\$20$ fixed cost and the firm decides it will limit the total cost to $\$3,040$. Find the maximum production.

2) (Minimizing Total Cost given the Production Level) Consider the Cobb-Douglas production function $Q = 2L^{.4}K^{.6}$ and the same cost function in problem 1. Minimize the total cost if 130 units are produced.

7.6 Related Rates and Volumes

Introduction

Along with max/min problems, related rates problems are one of the main applications introduced in calculus. The idea of related rates is that if two or more variables are related to each other (eg. radius r and area A of a circle), and one of the quantities is changing with time (for example, the radius $r(t)$ of a pond ripple may be expanding at a rate of $\frac{dr}{dt} = 1ft/sec$) than the other variable will also be changing with time (the area enclosed by the circular ripple is also changing with time.) Knowing the formula relating the area A to the radius r, namely

$$A = \pi r^2,$$

we may find the relationship between $\frac{dA}{dt}$ and $\frac{dr}{dt}$ uing the chain rule:

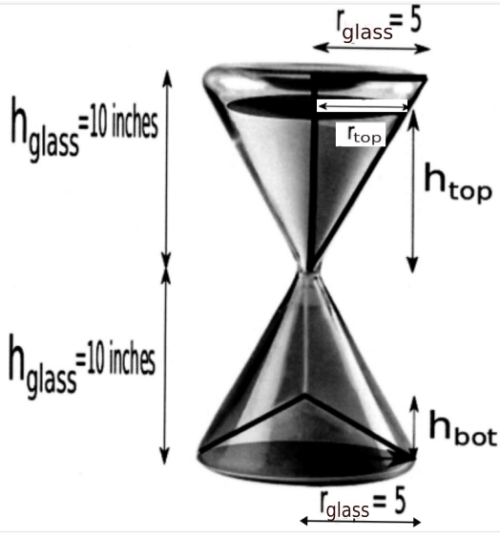
$$\frac{dA}{dt} = 2\pi r \frac{dr}{dt}.$$

Thus, for example, at the instant the radius of the ripple is $r = 10$, we can find the rate of change of the area:

$$\frac{dA}{dt} = 2\pi(10)(1)\, in^2/sec.$$

Example

Suppose sand is emptying from a conical hourglass in such a way that the height of the sand h_{top} in the top part is decreasing at a rate of $2\,in/min$. Assume the sand flows smoothly and forms a conical pile in the bottom part, how fast is the height h_{bottom} of the bottom conical pile increasing if the height of the sand in the top cone is 8 inches and the height of the sand in the bottom cone is 4 inches?



Solution.

1) Let r_{top} be the radius of the top conical pile of sand. Note by similar triangles

$$\frac{r_{top}}{h_{top}} = \frac{5}{10}$$

so we deduce that $r_{top} = \frac{1}{2}h_{top}$.

2) The volume of the top conical pile of sand is

$$V_{top} = \frac{1}{3}\pi r_{top}^2 h_{top}.$$

Substituting the relationship obtained in step 1), we have

$$V_{top} = \frac{1}{3}\pi(\frac{1}{2}h_{top})^2 h_{top} = \frac{\pi}{12}h_{top}^3.$$

3) Differentiating with respect to t the last equation relating V_{top} and h_{top}, we obtain

$$\frac{dV_{top}}{dt} = \frac{\pi}{4}h_{top}^2\frac{dh_{top}}{dt}.$$

4) Using the given information $h_{top} = 8\,in$ and $\frac{dh_{top}}{dt} = -2\,in/min$, we find

$$\frac{dV_{top}}{dt} = \frac{\pi}{4}16(-2) = -8\pi\,in^3/min.$$

5) Under the assumption that all the sand on top flows smoothly to the bottom conical pile, we have

$$\frac{dV_{bot}}{dt} = 8\pi\,in^3/min.$$

6) Note that

$$V_{bot} = \frac{1}{3}\pi(5^2)h_{bot} = \frac{25\pi}{3}h_{bot}.$$

7) Differentiating the last equation with respect to t gives

$$\frac{dV_{bot}}{dt} = \frac{25\pi}{3}\frac{dh_{bot}}{dt}.$$

8) Substituting the value in Step 5, we find

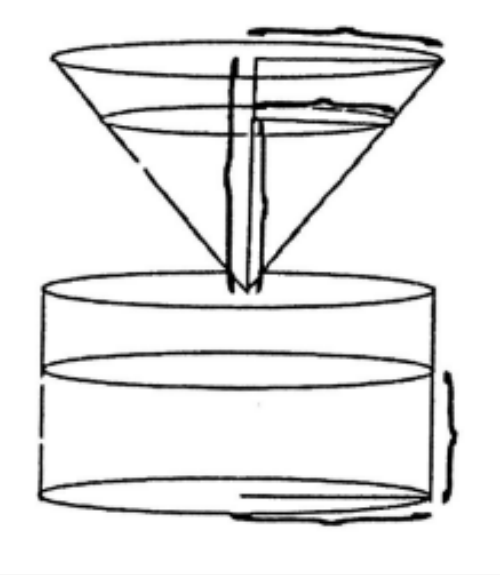
$$8\pi = \frac{25\pi}{3}\frac{dh_{bot}}{dt} \Rightarrow$$
$$\frac{dh_{bot}}{dt} = \frac{24}{25} in/min.$$

7.6.1 Exercise

Exercise

In the following problem, we will outline the steps used to analyze a hypothetical scenario where coffee drips from a filter in the shape of a right circular cone into a pot in the shape of a right circular cylinder. The radius of the filter and cylinder are both 3 inches and the height of the filter is 6 inches. If the level of the coffee in the cylindrical pot is rising at the constant rate of 1 inch per minute, how fast is the height of the coffee in the filter decreasing when the coffee in the filter is 5 inches deep?

1) On the diagram below, label the radius (3 in) of the filter and cylinder and the height (6 in) of the filter. Also, label the radius r_{cone} and height h_{cone} of coffee in the filter, and also the height h_{pot} of coffee in the pot.



2) Using similar triangles, express the radius r_{cone} in terms of the height h_{cone}.

3) Use your answer to problem 2 to give a formula for the volume V_{cone} of coffee in the filter in terms of h_{cone}.

4) Find the derivative $\frac{d}{dt}[V_{cone}]$ assuming that h_{cone} is a functions of time t.

5) Give a formula for the volume V_{pot} of coffee in the pot in terms of h_{pot}.

6) Find the derivative $\frac{d}{dt}[V_{pot}]$ assuming that h_{pot} is a function of time t.'

7) Give an equation that relates your answers to problem 4 and problem 6.

8) Answer the question posed at the beginning.

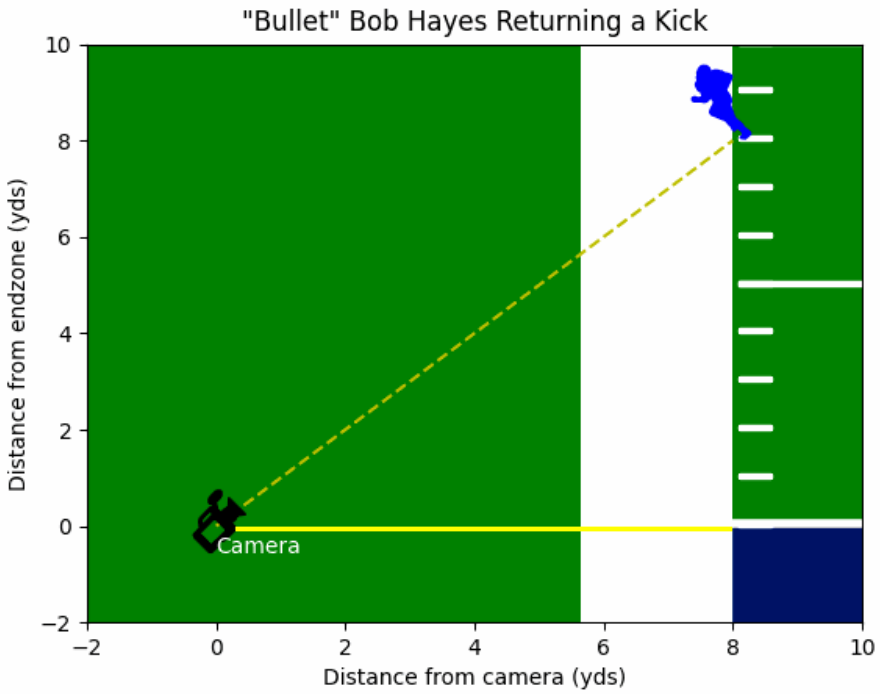
9) What assumption(s) have we made in analyzing this coffee pot?

7.7 Related Rates with Trig Functions and Football

In this lab, we will solve a related rates problem which involves the use of trig functions. In particular, we will determine how fast a camera is rotating as it tracks a football player speeding along the sideline to the endzone.

Note

Please visit the web version of this section to engage with this interactive resource.



The following code creates the animation, Football_Animation.gif. If you execute the code, you can open and view the .gif file.

Now the idea of the scenario is how we can use calculus to solve the problem of how fast the camera must rotate to stay pointed at the player. To begin, we need to review some basic info about trig functions.

7.7.1 Definition of Trig Functions

The acronym SOH-CAH-TOA is helpful to remember the definitions of the three basic trig functions sine, cosine, and tangent as ratios of sides of a right triangle:

$$\sin(\theta) = \frac{opposite}{hypotenuse}$$

$$\cos(\theta) = \frac{adjacent}{hypotenuse}$$

$$\tan(\theta) = \frac{opposite}{adjacent}$$

7.7.2 Derivatives of basic trig functions

The basic derivative formulas are

$$\frac{d}{d\theta}\sin(\theta) = \cos(\theta)$$

$$\frac{d}{d\theta}\cos(\theta) = -\sin(\theta)$$

$$\frac{d}{d\theta}\tan(\theta) = \sec^2(\theta) = \frac{1}{\cos^2(\theta)}$$

In our application problem, we will assume that the angle θ is changing with time t. In this case, the derivatives with respect to time t are given by

$$\frac{d}{dt}\sin(\theta(t)) = \cos(\theta)\frac{d\theta}{dt}$$

$$\frac{d}{dt}\cos(\theta(t)) = -\sin(\theta)\frac{d\theta}{dt}$$

$$\frac{d}{dt}\tan(\theta(t)) = \sec^2(\theta)\frac{d\theta}{dt} = \frac{d\theta/dt}{\cos^2(\theta)}$$

Example

Find $\frac{d}{dt}$ of $\tan(t^2)$. (Hint: In this case, $\theta(t) = t^2$)

Answer: $\sec^2(t^2)2t$ or $\frac{2t}{\cos^2(t^2)}$

Explanation:

It's been given that the derivative d/dt of $\tan(\theta(t))$ is $\sec^2(\theta)\frac{d\theta}{dt}$. Therefore, since $d\theta/dt$ is $2t$, multiplying by $\sec^2(\theta)$ gives $\sec^2(t^2)2t$ or equivalently, $\frac{2t}{\cos^2(t^2)}$.

7.7.3 Radian Measure

The above formulas for the derivatives of $\sin(\theta)$, $\cos(\theta))$ and $\tan(\theta)$ all assume that the angle θ is measured in radians.

Conversion between radians and degrees is possible if we remember that π radians = 180 degrees or 2π radians = 360 degrees.

Example

Convert $\frac{5}{6}$ radians to degrees

Answer: $\frac{5}{6}\ radians$ x $\frac{180\,degrees}{\pi\,radians}$ = $\frac{150}{\pi}$ degrees

Explanation:

It's been given that π radians = 180 degrees so $\frac{180\,degrees}{\pi\,radians} = 1$. We can complete the conversion using cross cancellation method.

7.7.4 Python trig functions

```
from math import pi, sin, cos, tan, asin, acos, atan, radians, degrees
```

You can use these imported python functions to do calculations within the notebook. For correct syntax, code as follows:

```
sin(pi/4)
```

```
0.7071067811865475
```

To convert from radians to degrees, simply type degrees(radian value). This will give you the corresponding degree value:

```
degrees(pi)
```

```
180.0
```

Furthermore, by default Python assumes the input angle measure is in radians. You can convert the final answer to degrees as such:

```
degrees(atan(1))
```

```
45.0
```

Note that the above example asks for the degree conversion of the radian value of the arctangent of a right triangle with congruent sides (8/8, 6/6, and so on).

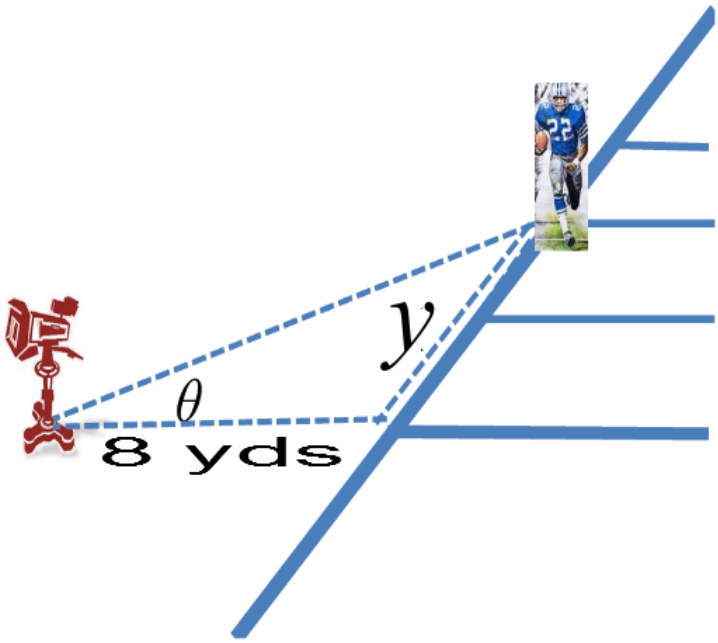
Example

Bob Hayes won the 100 meter dash at the Tokyo '64 Olympics equaling the world record of 10.0 seconds. Hayes went on to be a star wide receiver with the Dallas Cowboys.

Suppose a TV camera is positioned on the goal line and 8 yards from the sideline. The camera stays fixed on Hayes, who is currently on the 8 yard line and sprinting towards the end zone at 10 yards per second.

Answer the questions below to determine how fast the camera is turning (in degrees per second).

1) From the Figure below , give an equation for y in terms of θ.



Answer: $y(t) = 8\tan(\theta(t))$

Explanation:

We know that $\tan(\theta) = \frac{opposite}{adjacent}$.

In our case, the opposite side is y and adjacent side is 8, so $\tan(\theta) = \frac{y}{8}$.

To convert to $y(t)$, we multiply on both sides and get $y(t) = 8\tan(\theta)$

2a) What is the physical meaning of $\frac{dy}{dt}$?

Answer: The physical meaning of $\frac{dy}{dt}$ is the speed of Bob Hayes as he is running towards the end line from the side line measured in yds/sec. A negative value indicates he is getting closer to the goal line.

2b) What is the physical meaning of $\frac{d\theta}{dt}$?

Answer: The physical meaning of $\frac{d\theta}{/dt}$ is how fast the camera angle is turning as it keeps pointing at Hayes as he runs towards the goal line. A negative value indicates clockwise rotation.

3) If θ is measured in radians, then $\frac{d}{dt}\tan(\theta(t)) = \frac{1}{\cos^2\theta}\frac{d\theta}{dt}$. Use your answer to problem 1 to find a general relationship between $\frac{dy}{dt}$ and $\frac{d\theta}{dt}$.

Answer: If $y = 8\tan(\theta)$, then $dy/dt = 8\frac{d\theta/dt}{\cos^2(\theta)}$

Explanation:

We learn that the derivative of $tan(\theta) = \frac{1}{\cos^2\theta}$.

Using chain rule, we deduce that the derivative of $tan(\theta(t)) = \frac{1}{\cos^2\theta}\frac{d\theta}{dt}$.

Thus, $\frac{dy}{dt}$, using the equation in question one in which $y(t) = 8\tan(\theta(t))$, equals $8\frac{d(\theta)/dt}{\cos^2(\theta)}$

4) Find $\frac{d\theta}{dt}$ (in radians per second) at the instant when $y = 8$. (Hint: $\cos(\theta) = \frac{adjacent}{hypotenuse}$).

   Answer: $-\frac{5}{8}$

Explanation:

$$y(t) = 8\tan(\theta(t)) \Rightarrow \tan(\theta(t)) = \frac{y(t)}{8}.$$

Since $y(t) = 8$, then $\tan(\theta(t)) = 1$, which means $\theta(t)$ is $\frac{\pi}{4}$ radians .

The formula

$$\frac{dy}{dt} = \frac{\frac{d\theta}{dt}}{\cos^2(\theta)}$$

implies that

$$\frac{d\theta}{dt} = \frac{\cos^2(\theta)\frac{dy}{dt}}{8}.$$

We can substitute θ with $\frac{\pi}{4}$ and $\frac{dy}{dt}$ with -10.

Then we get:

$$-\frac{10\cos^2(\frac{\pi}{4})}{8} = -\frac{5}{4}\cos^2(\frac{\pi}{4}) = -\frac{5}{4}(\frac{\sqrt{2}}{2})^2 = -\frac{5}{4} * \frac{1}{2} = -\frac{5}{8}.$$

5) Use the fact that 1 radian = $(\frac{180}{\pi})^o$ to express your answer to problem 4 in degrees per second.

   Answer: $-\frac{225}{2\pi}$ degrees per second or around 35.8 degrees per second.

Explanation:

Multiply $-\frac{5}{8}\frac{radians}{second}$ by $\frac{180\,degrees}{\pi\,radians} = \frac{225\,degrees}{2\pi\,seconds}$ = -35.8 degrees per second.

6) Let R be your answer to problem 5. If the camera revolves at the constant rate of R degrees per second, how many seconds would it take to make one 360° revolution?

   Answer: It will take the camera approximately 10 seconds to make a full revolution.

Explanation:

Using the cross-canceling unit method, $(360degrees)\frac{1\ second}{R\ degrees}$, we cancel the degrees and we are given seconds: $360/35.8 = 10.04$ seconds.

Questions

1. Find θ in degrees if the camera was 3.5 yds from the sidelines.
2. If the camera is rotated .5 radians away from the endzone towards Bob Hayes, what yard line must he be on? (Given he is running at velocity of 10yds/sec again).
3. Find θ in radians when t = 0.4 seconds.
4. Find $\frac{d\theta}{dt}$ in degrees/sec if Bob Hayes had pulled his hamstring and could only run 6 yds/sec at the instant when y=5.
5. Find $\frac{d\theta}{dt}$ in radians/sec of the angle relative from Bob to the camera when y=4 and he was going 9 yds/sec.

7.7.5 Exercises

Hint: draw pictures to assist you in visualizing the problem and marking necessary measurements.

Exercises

1. A baseball diamond is a square with side length of 90ft. A batter hits the ball and runs toward first base with a speed of 20ft/s. At what rate is his distance from second base decreasing when he is halfway to the first base? At what rate is his angle to second base changing?
2. At 1:00 PM, Jet-ski A is 7 miles east of Jet-ski B. Jet-ski A is moving north at 20 miles/hr while Jet-ski B is moving south 30 miles/h. How fast is the distance between the Jet-skis changing at 3:00 PM?

7.8 Probability Distributions and Drive Thrus

7.8.1 Introduction

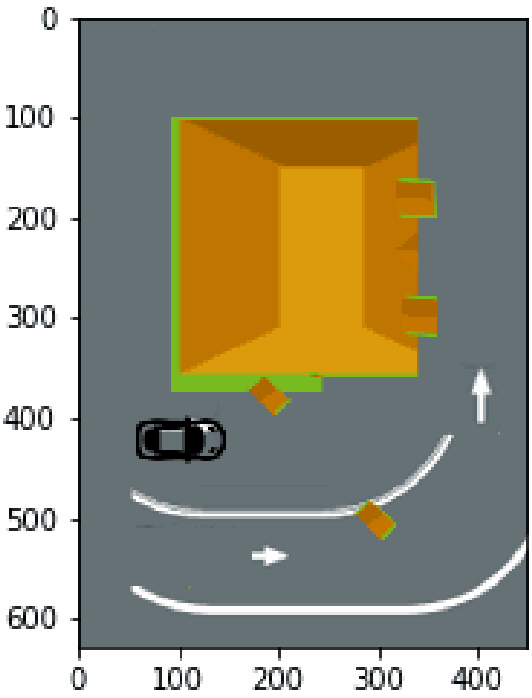
Introduction

In this section, we show how random numbers are used to simulate a fast-food restaurant with a drive-thru order window as illustrated in the animation below.

Note

Please visit the web version of this section to engage with this interactive resource.

For an explanation of how to create this animation, see the "Introduction to Python" section in the Python Programming Guide



For Discussion

Suppose you are the owner of a fast-food restaurant. Currently, there is one drive-thru lane, but you are considering adding a second drive-thru lane. What factors need to be considered in deciding whether to add a second drive-thru lane?

A statistical simulation model uses data of customer arrival and ordering times to re-create a scenario and explore options such as adding a second drive-thru lane. (For example, see the video

"Fast Food Drive Thru 2 lanes versus 1" YouTube, uploaded by Daniel Hickman, 5 March 2013, https://www.youtube.com/watch?v=M9v6bx3l4Dk. Permissions: YouTube Terms of Service

Calculus is used in statistically-based simulations since probabilities are represented as areas under curves called probability distributions.

Here we will focus on the use of uniform (constant) and exponential distributions to model arrival times of customers.

7.8.2 Distributions

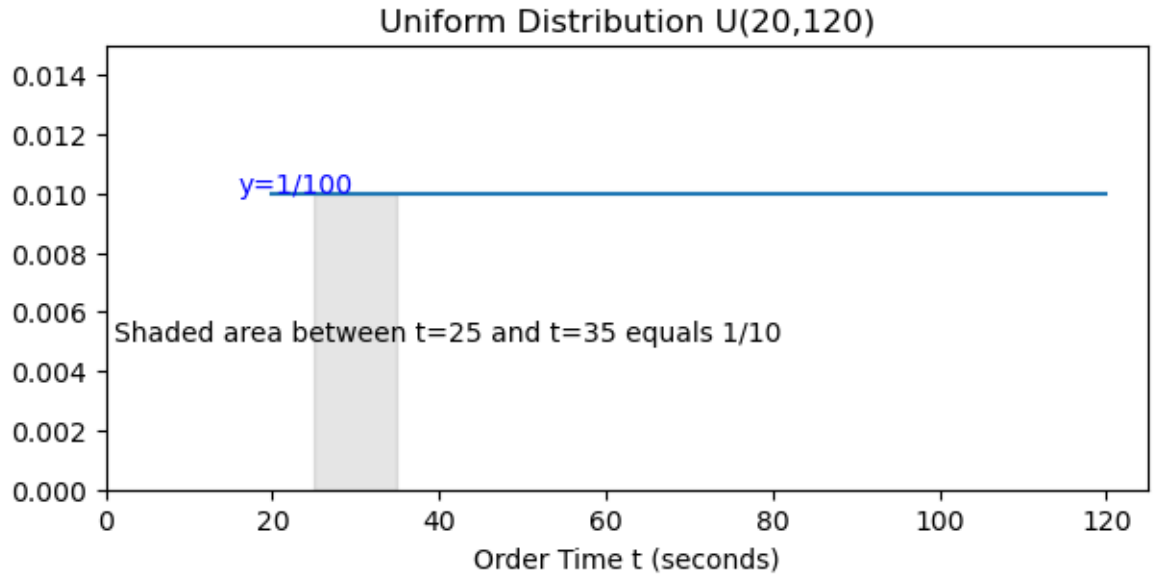
A computer simulation of 1 versus 2 drive-thru lanes uses probabilities computed using definite integrals. The probability that some timed event X occurs between $t = a$ and $t = b$ is given by

$prob(a \leq X \leq b) = \int_a^b f(t)\, dt$

Example 1 Suppose the time to place an order is given by a uniform distribution $U(20, 120)$. In this case, $f(t) = \frac{1}{100}$ where $20 \leq t \leq 120$. Find the probability that the length of the next order is between 25 and 35 seconds.

Solution: probability=$\int_{25}^{35} f(t)dt = \int_{25}^{35} \frac{1}{100}dt = \frac{t}{100} \,|_{25}^{35} = \frac{35}{100} - \frac{25}{100} = \frac{10}{100} = \frac{1}{10}$.

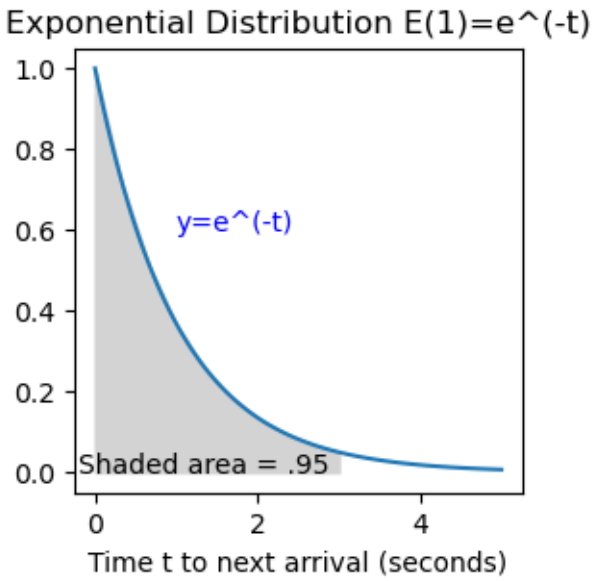
```
Text(0.5, 0, 'Order Time t (seconds)')
```



Example 2a) An exponential distribution has the form $f(t) = ke^{-kt}$ $(0 \leq t \leq \infty)$ where the parameter k is such that the mean is $1/k$. Suppose the time to the next car arriving at the drive-thru is given by E(1) (i.e. k=1 so the mean is 1 minute). What is the probability that the next arrival is within 3 minutes?

Solution. probability = $\int_0^3 e^{-t}\, dt = -e^{-t} \,|_0^3 = -e^{-3} + e^0 = 1 - e^{-3} \approx .95$.

```
Text(0.5, 0, 'Time t to next arrival (seconds)')
```



Example 2b) Find the value of k so that there is a 25% chance (probability=.25) that the next car will arrive within 1 minute.

Solution $\int_0^1 ke^{-kt}\,dt = .25 \Rightarrow -e^{-kt}\,|_0^1 = .25 \Rightarrow -e^{-k} + 1 = .25 \Rightarrow -e^{-k} = -.75 \Rightarrow e^{-k} = .75$. Take ln() of both sides, we have $-k = \ln(.75) \Rightarrow k = -\ln(.75) \approx .29$.

Example 3a) Using the value of k given in 2b), find the probability $F(x)$ that the next car will arrive within x minutes.

Solution $F(x) = \int_0^x .29e^{-.29t}\,dt = -e^{-.29t}\,|_0^x = -e^{-.29x} + 1$. That is $F(x) = 1 - e^{-.29x}$

Example 3b) Use your function in part a) to find the probability that the next car will arrive within 2 minutes.

Solution
$F(2) = 1 - e^{-.29(2)} = 1 - e^{-.58} \approx .44$

7.8.3 Use of Random Numbers

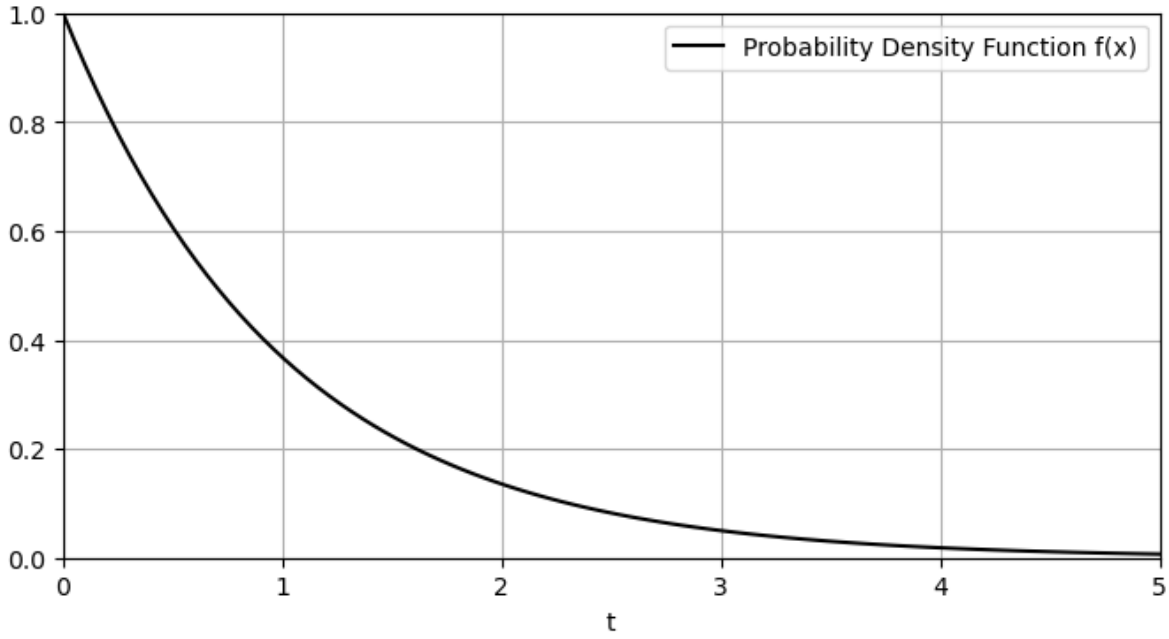
We will explain how a value r generated by a PRNG (pseudo-random number generator) can be converted into values of random variables such as

- A=length of time before the next car arrives at the parking lot
- O=length of time to place an order
- P=length of time to pay for the food
- F=length of time to pick up food

A=Length of time before next arrival

Let X be a random variable giving the length of time (in minutes) until the next car arrives at the parking lot. We assume X is an exponential distribution with mean $1/k = 1$. In this case, the probability density function (pdf) is

$$f(x) = ke^{-kx} = e^{-x}$$



The corresponding cumulative distribution function (cdf) is

$$F(t) = prob(0 \le X \le t) = \int_0^t ke^{-kx}\,dx = -e^{-kx}\,|_0^t = -e^{-kt} + 1 = -e^{-t} + 1.$$

Here is a graph of this cdf:

Note that the $F(t) = 1 - e^{-t}$ is an increasing function with a value of 0 at $t = 0$ and a range $[0, 1)$. As such, each random number r $(0 \le r < 1)$ corresponds to a unique value of A:

$$r = 1 - e^{-kA} \;\Rightarrow$$

$$e^{-kA} = 1 - r \;\Rightarrow$$

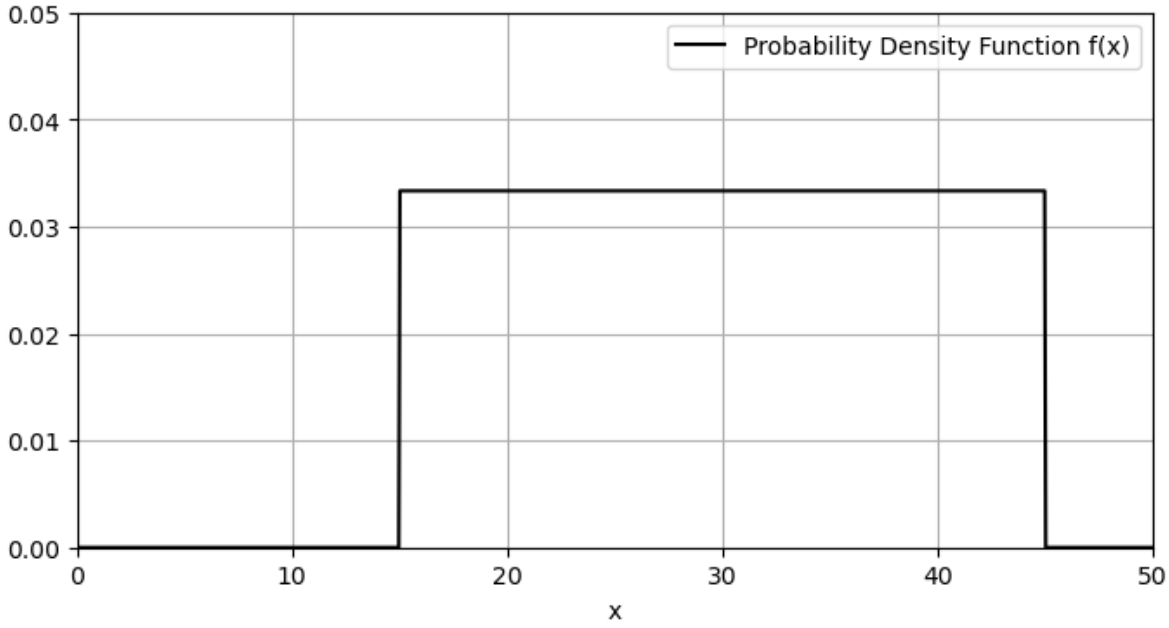
$$-kA = \ln(1 - r) \;\Rightarrow$$

$$A = -\frac{1}{k}\ln(1 - r) = -\ln(1 - r).$$

O=length of time to place an order

Let X be a random variable giving the length of time (in seconds) to place an order. We assume X is uniformly distributed on the interval [15,45] (denoted U(15,45)) In this case, the probability density function (pdf) is

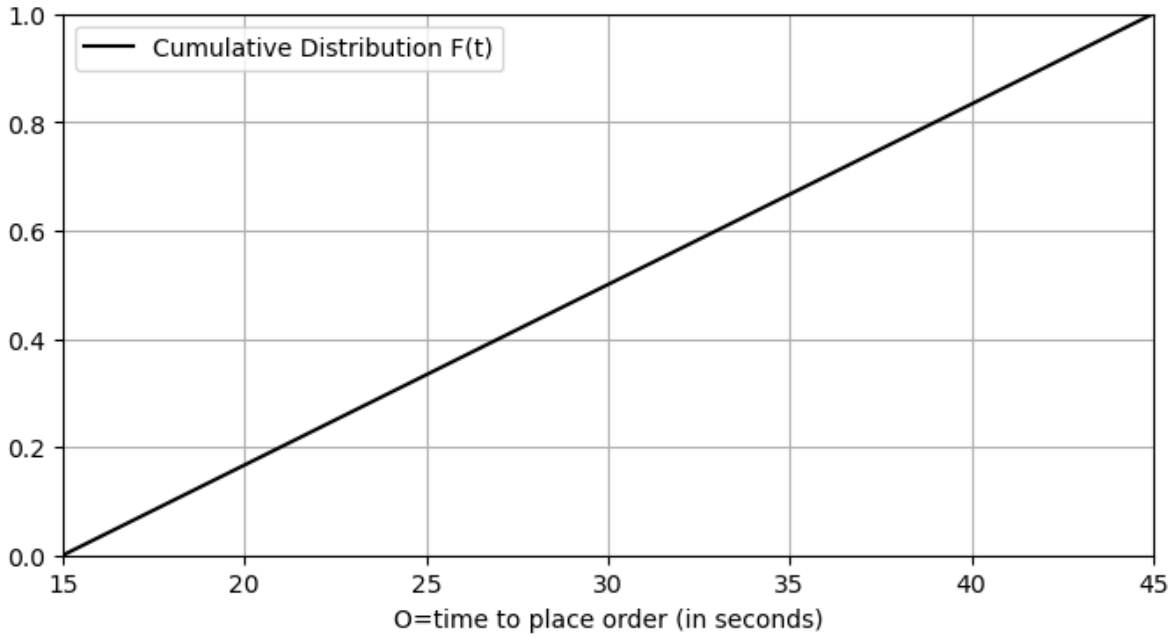
$$f(x) = \frac{1}{30} \ (15 \leq x \leq 45)$$



Then, the cumulative distribution function (cdf) is

$$F(t) = prob(15 \leq X \leq t) = \int_{15}^{t} \frac{1}{30}\, dx = \frac{1}{30} x \,|_{15}^{t} = \frac{t}{30} - \frac{15}{30} = \frac{t-15}{30}$$

Here is a graph of this cdf:

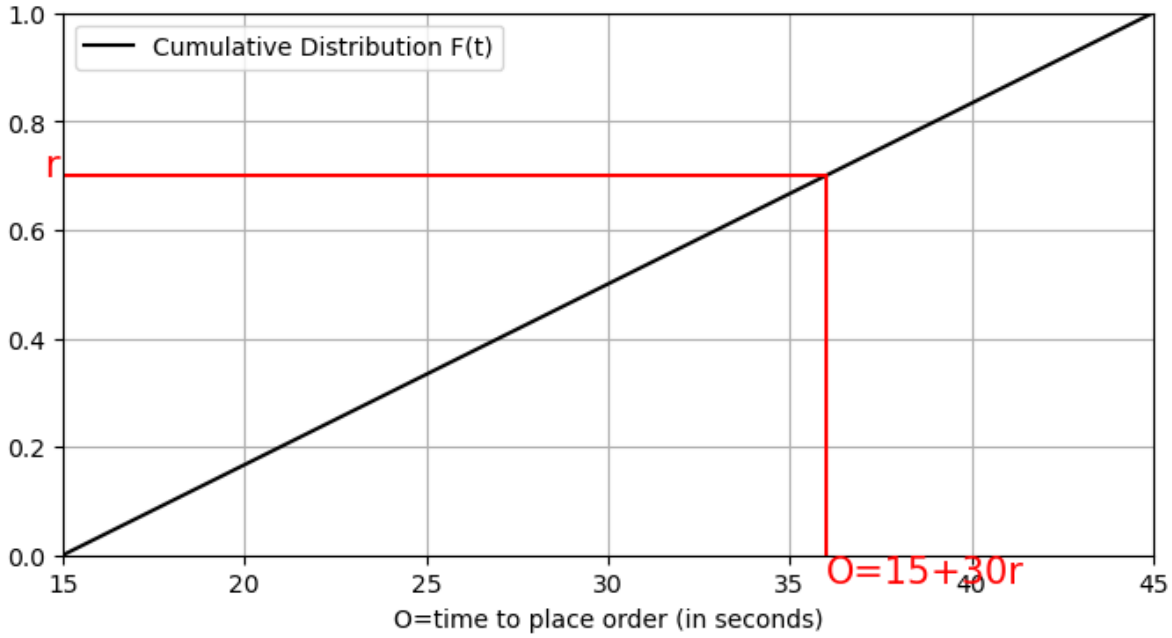


As before, the $F(t) = \frac{t-15}{30}$ $(15 \leq t \leq 45)$ is an increasing function with a value of 0 at $t = 15$ and a range $[0, 1)$. As such, each random number r $(0 \leq r < 1)$ corresponds to a unique value of O where $15 \leq O \leq 45$:

$$r = \frac{O - 15}{30} \Rightarrow$$

$$O - 15 = 30r \Rightarrow$$

$$O = 15 + 30r.$$



7.8.4 Exercises

Exercises

1. An important probability function called the exponential distribution has the form

$$f(t) = ke^{-kt} \quad (t \geq 0 \quad and \quad k > 0).$$

The exponential distribution is used to describe the amount of time until some event occurs such as the time to the next arrival of a car at a drive-thru window. More specifically, the probability that the next car arrives somewhere between $t = a$ and $t = b$ minutes is given by the definite integral

$$Prob(a \leq t \leq b) = \int_a^b f(t)dt = \int_a^b ke^{-kt}\ dt.$$

a) Suppose $k = 1$. Find the probability that the next car will arrive at the drive-thru within 2 minutes.

b) Using $k = 1$, show that the probability that the next car will arrive between 0 and ∞ minutes is equal to 1 (i.e. there is a 100% chance another car will arrive sometime in the future).

c) Find the value of k such that there is a 50% chance (i.e. probability = .5) that the next car will arrive in one minute.

d) Using your answer to part c), find a function $F(x)$ which gives the probability that the next car will arrive within x minutes. (In general, given a probability distribution function $f(t)$ defined on an interval $a \leq x \leq b$, the cumulative distribution function $F(x)$ is obtained by the definite integral $\int_a^x f(t) \quad dt$).

e) Use your answer to d) to find the probability that the next car will arrive within 3 minutes.

2. Let $P =$ length of time to pay for an order of food.

a) Let X be a random variable giving the length of time (in seconds) to pay for an order using cash. We assume X is uniformly distributed on the interval [5,35] (denoted U(5,35)). Explain how to use a pseudo-random number to give a value for P.

b) Let X be a random variable giving the length of time (in seconds) to pay for an order using a credit card. We assume X is uniformly distributed on the interval [5,20] (denoted U(5,20)). Explain how to use a pseudo-random number to give a value for P.

3. Let F be the length of time to pick up food. Let X be a random variable giving the length of time (in seconds) to pick up an order. We assume X is uniformly distributed on the interval [30,90] (denoted U(30,90). Show how to use a pseudo-random number to generate a value for F

7.9 Normal Distribution and Process Control

In this lab, we will apply the Normal distribution to statistical process control. The main goal of the latter is to ensure the output of a manufacturing process remains within limits determined by usual random variation. If a very low probability event occurs, this may be used as a signal that there is a problem with the manufacturing process, in which case the process might be stopped and equipment checked for possible defects.

7.9.1 Derivatives of Exponential Functions

The normal distribution is given by

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

where μ is a constant called the *mean* and σ is another constant called the *standard deviation.*

To analyze the normal distribution using calculus, we need the derivative formula

$$\frac{d}{dx} e^{u(x)} = e^{u(x)} u'(x).$$

For example, let $f(x) = c_1 e^{-\frac{(x-c_2)^2}{2c_3^2}}$. Then,

$$f'(x) = c_1 e^{-\frac{(x-c_2)^2}{2c_3^2}} [\frac{-2(x-c_2)}{2c_3^2}] = c_1 e^{-\frac{(x-c_2)^2}{2c_3^2}} [\frac{-(x-c_2)}{c_3^2}]$$

Note that $f'(c_2) = 0$. Since $f'(x) > 0$ (f is increasing) for all $x < c_2$ and $f'(x) < 0$ (f is decreasing) for all $x > c_2$, we know $f(x)$ has an absolute maximum value at $x = c_2$.

We use the product rule to find $f''(x)$:

$$f''(x) = c_1 e^{-\frac{(x-c_2)^2}{2c_3^2}} [\frac{-(x-c_2)}{c_3^2}]^2 + c_1 e^{-\frac{(x-c_2)^2}{2c_3^2}} (-\frac{1}{c_3^2})$$

$$= c_1 e^{-\frac{(x-c_2)^2}{2c_3^2}} [\frac{(x-c_2)^2}{c_3^4} - \frac{1}{c_3^2}].$$

To find the x-coordinates of the two inflection points, set $f''(x) = 0$:

$$\frac{(x-c_2)^2}{c_3^4} - \frac{1}{c_3^2} = 0 \Rightarrow$$

$$\frac{(x-c_2)^2}{c_3^4} = \frac{1}{c_3^2} \Rightarrow$$

$$(x-c_2)^2 = c_3^2 \Rightarrow x - c_2 = \pm c_3 \Rightarrow$$

$$x = c_2 \pm c_3.$$

The inflection points occur symmetrically about the maximum point at $x = c_2$.

7.9.2 Rule of 68-95-99.7

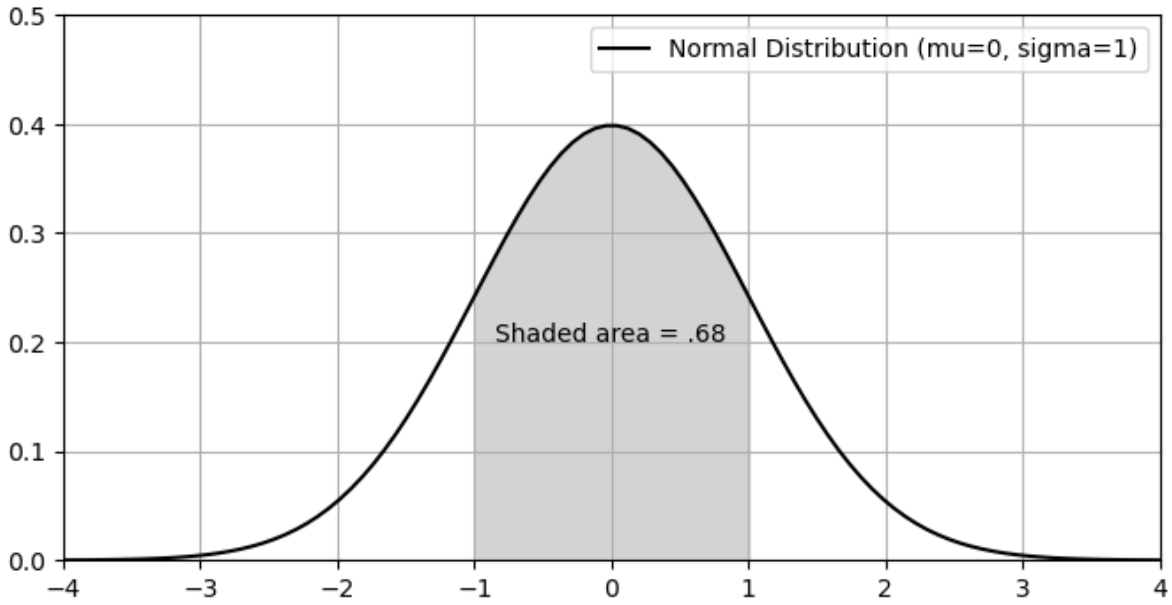
If data points are normally distributed,

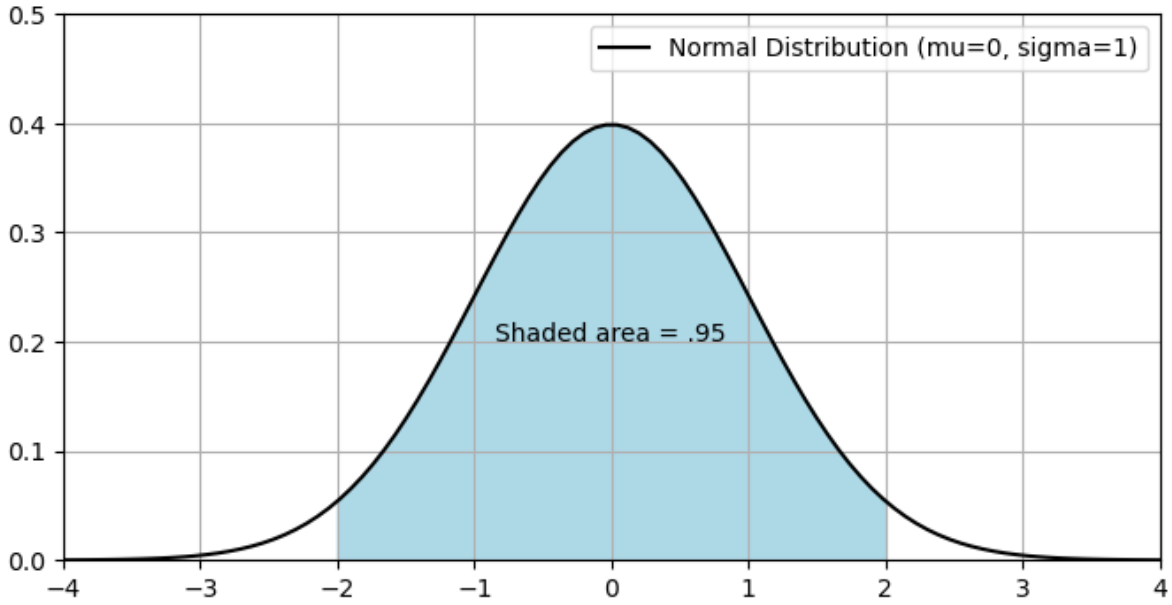
68% of the data will be within 1 standard deviation of the mean,

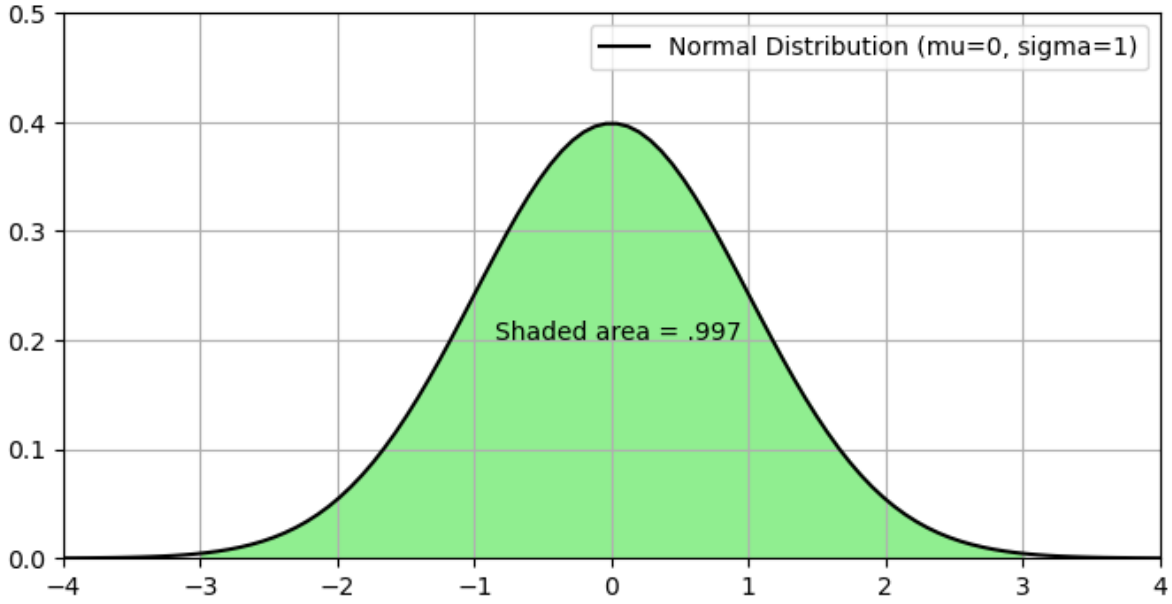
95% of the data will be within 2 standard deviations of the mean, and

99.7% of the data will be within 3 standard deviations of the mean.

The plot below shows the standard normal distribution with mean $\mu = 0$ and standard deviation $\sigma = 1$. The total area under the curve is equal to 1. The shaded area on the graph below represents the probability that standard normally distributed data will be within 1 standard deviation of the mean.







7.9.3 Exercises

Exercises

Suppose a manufacturing process of a certain part is "in control" when a sample is normally distributed with average length $\mu = 1$ inch (mean) and standard deviation $\sigma = .01$ inch.

1) Make a sketch of a normal distribution with this mean and standard deviation, showing the rule of 68-95-99.7.

2) Answer the following:

a) Out of 100 samples, how many would we expect to have an average length between .997 and 1.003 inches?

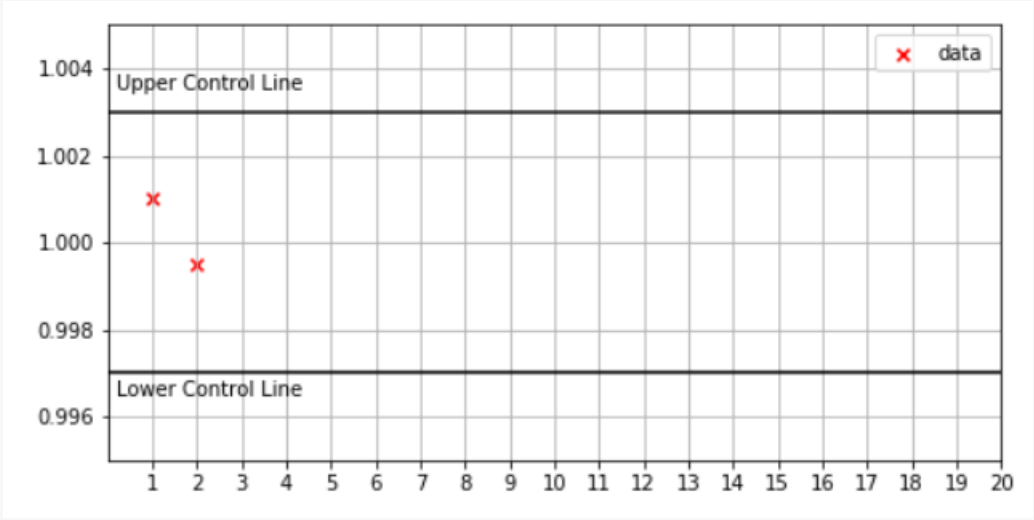
b) Out of 100 samples, how many would we expect to have an average length of less than .997 inches?

c) Out of 100 samples, how many would we expect to have an average length of less than 1.003 inches?

3) Consider the following sequence of twenty average sample lengths:

1.001, .9995, 1.000, 1.001, 1.002, .998, .999, 1.0025, .9975, .9825, 1.0025, 1.00075, 1.000, .999, .9985, 1.0035, 1.001, 1.000, 1.0015

a) Complete the control chart corresponding to these sample sizes. The outer boundary lines are within 3 standard deviations of the mean. If a data point goes outside these boundaries, this signals the process may be out of control.



b) Suppose the process is stopped if the average sample length has a probability of less than .0015 of occurring. Will the process be stopped? If so, at what point(s)?

4) A normal distribution with mean μ and standard deviation σ is given by

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where μ is the mean and σ the standard deviation $\sigma > 0$

Use the first and second derivatives to find the exact coordinates of the max/min and inflection points of this exponential function if $\mu = 1$ and $\sigma = 0.01$.

7.10 Partial Derivatives and OLS Regression

Ordinary least squares (OLS) regression of 2-dimensional data is an important concept in data analysis. Given a collection of data points

$$(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$$

the goal is to find a line $y = mx + b$ (called the OLS regression line) that best fits the data in the sense that it minimizes the sum of square vertical separations between the data points and the line:

$$S(m, b) = (y_1 - (mx_1 + b))^2 + (y_2 - (mx_2 + b))^2 + ... + (y_n - (mx_n + b))^2.$$

The optimal values of m and b minimize $S(m, b)$, and hence may be obtained using a system of two equations and 2 unknowns:

$$\frac{\partial S}{\partial m} = 0$$

$$\frac{\partial S}{\partial b} = 0$$

Example

Consider the data values (1,1), (2,3), and (4,3). In this case,

$$S(m,b) = (m+b-1)^2 + (2m+b-3)^2 + (4m+b-3)^2.$$

This gives rise to the system

$$\frac{\partial S}{\partial m} = 2(m+b-1) + 2(2m+b-3)(2) + 2(4m+b-3)(4) = 42m + 14b - 38 = 0$$

$$\frac{\partial S}{\partial b} = 2(m+b-1) + 2(2m+b-3) + 2(4m+b-3) = 14m + 6b - 14 = 0.$$

The system is equivalent to

$$21m + 7b = 19$$

$$7m + 3b = 7.$$

We can solve this system by multiplying the first equation through by 3, and the second equation through by 7

$$63m + 21b = 57$$

$$49m + 21b = 49.$$

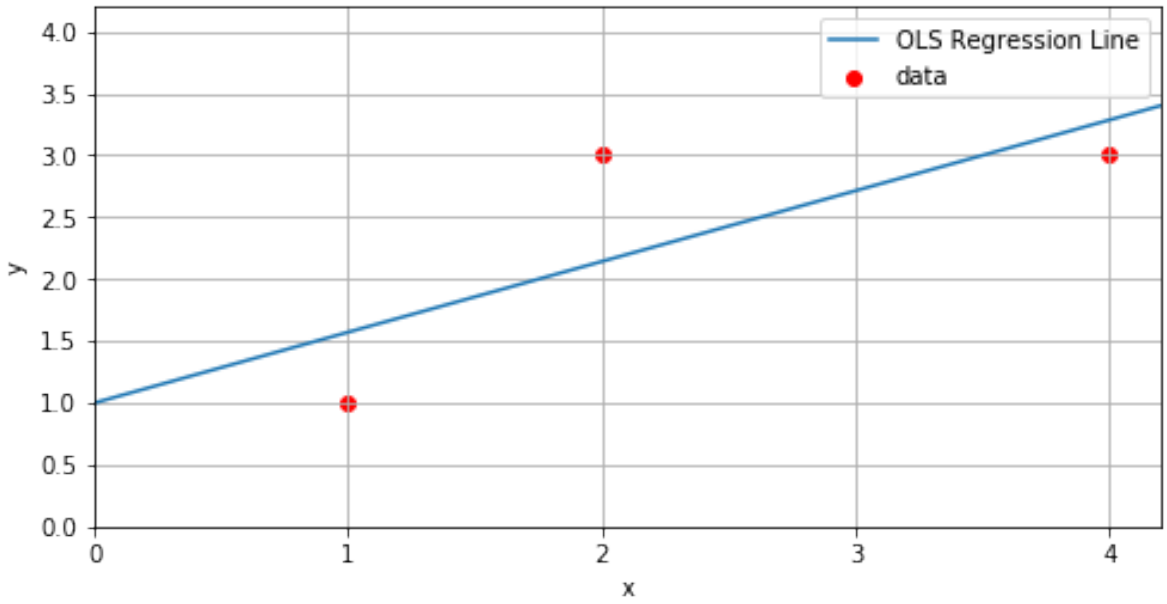
Subtracting the second equation from the first equation gives

$$14m = 8 \Rightarrow m = \frac{4}{7}.$$

We find b by back substitution:

$$7(4/7) + 3b = 7 \Rightarrow 3b = 3 \Rightarrow b = 1.$$

The plot below shows the OLS regression line $y = \frac{4}{7}x + 1$ together with the three data points.



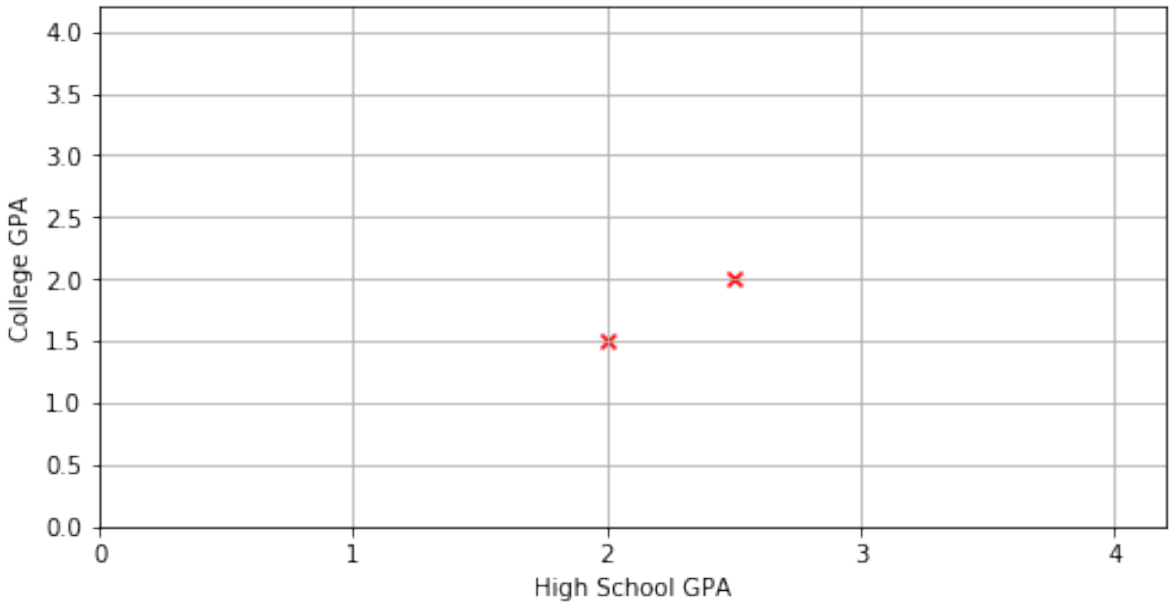
7.10.1 Exercises

Exercises

A college admissions officer has compiled the following data relating 8 students' high school and college GPA:

GPA								
High School GPA:	2.0	2.5	3.0	3.0	3.5	3.5	4.0	4.0
College GPA:	1.5	2.0	2.5	3.5	2.5	3.0	3.0	3.5

1) Complete the plot of this data and then guess the value of m and b for the least squares regression line $y = mx + b$. (Use your guess to check that your answer to problem 2 is reasonable).



Exercises (continued)

2) Use calculus to find the least squares regression line $y = mx + b$.

3) Use the second derivative test to verify the choice of m and b in problem 2 that gives a minimum for the sum of squares $S(m, b)$.

4) Use the regression line to predict the college GPA's:

GPA			
High School GPA:	2.0	3.0	4.0
College GPA:			

7.11 Area Between Curves and the Gini Index

Areas under curves called probability distributions can be used to represent probabilities. Another way to use area is to represent an inequitable distribution of resources as the area between two curves.

7.11.1 Computing Area Between Two Curves

Suppose $f(x) \leq g(x)$ on an interval $a \leq x \leq b$. Then the area A between the graph of $y = f(x)$ and $y = g(x)$ on the interval $a \leq x \leq b$ is given by the definite integral.

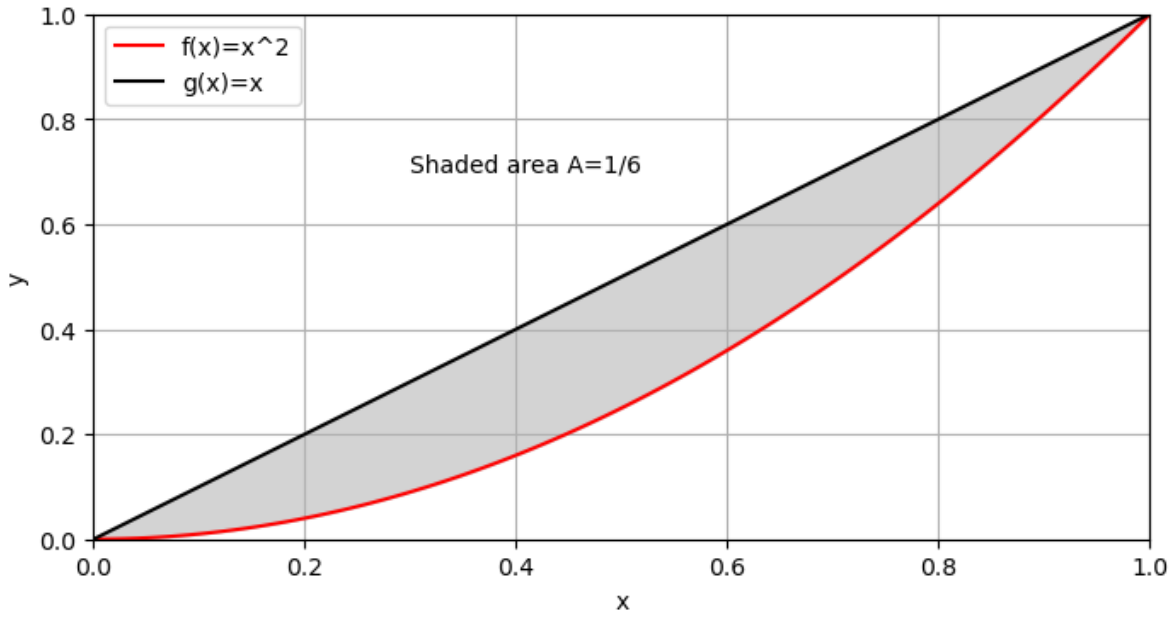
$$A = \int_a^b [g(x) - f(x)]\, dx.$$

Example

Find the area A between $f(x) = x^2$ and $g(x) = x$ on the interval $0 \leq x \leq 1$.

Solution.

$$A = \int_0^1 (x - x^2)\, dx = \frac{x^2}{2} - \frac{x^3}{3} \,|_0^1 = \frac{1}{2} - \frac{1}{3} - (0 - 0) = \frac{1}{6}.$$

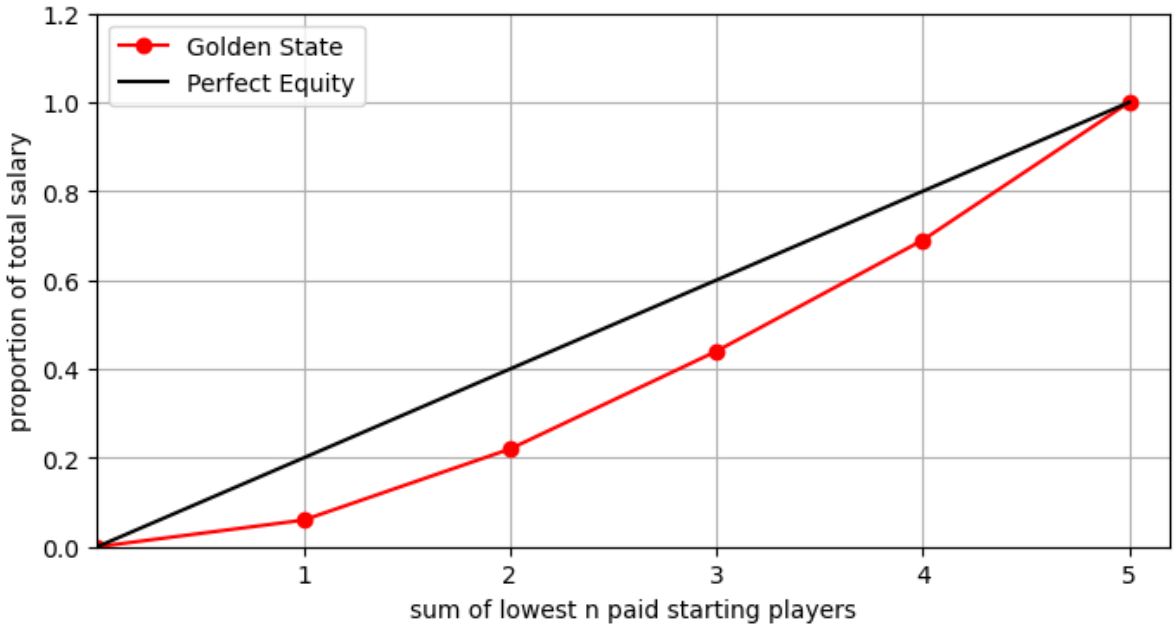


7.11.2 Lorentz Curve and GINI Index

Consider the following NBA salary data:

Player	Salary
S Curry	45,780,966
K Thompson	37,980,720
A Wiggins	31.579.390
D Green	24,026,712
J Wiseman	9,166,800

The salaries are clearly not evenly distributed between the five starting players. To measure the degree of uneven distribution, we first make a plot of what is known as a *Lorentz curve*.



The Lorentz curve plotted in red in this case tells us that the lowest paid starter received .06 = 6% of the total starting 5 salary. This is indicated by the red point at (1,.06). The 2 lowest paid starters received .22 of the salary total. This is indicated by the red point at (2, .22). The 3 lowest paid starters received .44 of the total. This is indicated by the red point at (3, .44). The 4 lowest paid starters received .69 of the total. This is indicated by the red point at (4, .69). The final point must be (5,1) since the 5 starters account for 100% of the total starting salary.

Note that the least inequity occurs when the Lorentz curve coincides with the line $y = x$. In this case, the area A between $y = x$ and the Lorentz curve is equal to zero ($A = 0$).

The greatest inequity occurs if one person receives everything and the rest nothing. In this case, the area A between $y = x$ and the Lorentz curve is one-half ($A = 1/2$).

The GINI index G.I. measures iniquity as $G.I. = 2A$ so that $0 \leq G.I. \leq 1$.

The larger the value of the GINI index, the greater the inequity.

7.11.3 Exercises

Exercises

Suppose the Lorentz curve for Profession 1 is $L_1(x) = x^{1.7}$ and the Lorentz curve for Profession 2 is $L_2(x) = .8x^2 + .2x$.

1) Compute the GINI index of both profession 1 and profession 2 to determine which has the more even income distribution.

2) Plot the Lorentz curve for each profession and shade the area A between the curve and the line $y = x$ for each profession.

7.12 Integral Test and Income Streams

In this lab, we show that it is possible due to continuous compounding of interest to deposit a fixed amount P at time $t = 0$, and then withdraw at times $t = 1, 2, 3, ...$ increasing amounts $A + Bt$ (where $A > 0$ and $B > 0$ are constants) "in perpetuity" while always maintaining a positive balance in the account.

7.12.1 Integral Test

The integral test is one of the standard tests for convergence/divergence of an infinite series. The basic idea of the test is that for a continuous function $f(x)$ defined on the interval $1 \leq x < \infty$, the infinite series

$$\sum_{n=1}^{\infty} f(n)$$

and the improper integral

$$\int_{x=1}^{\infty} f(x)\, dx$$

either both converge or both diverge.

7.12.2 P-Series

The integral test can be used to determine all positive values of p for which the p series

$$\sum_{n=1}^{\infty} \frac{1}{n^p}$$

converges or diverges.

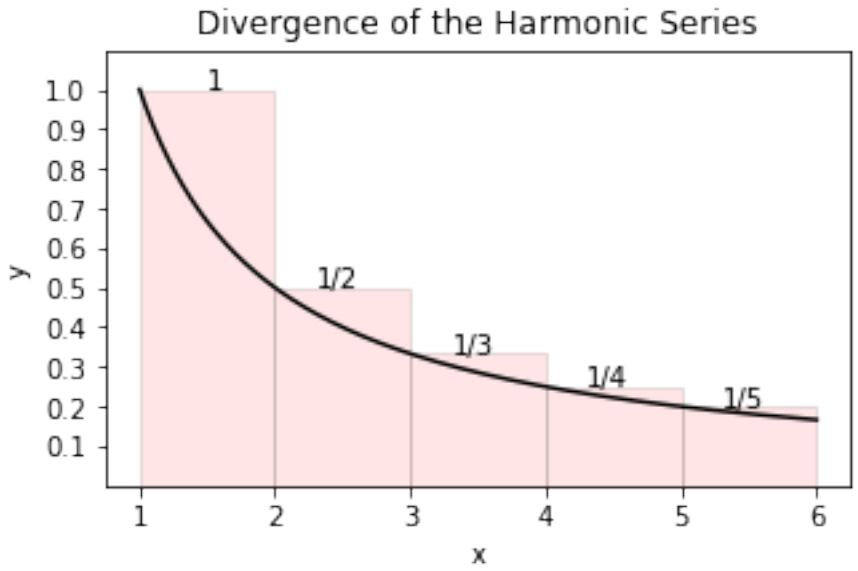
For example the case $p = 1$ so we have the harmonic series

$$\sum_{n=1}^{\infty} \frac{1}{n}.$$

The corresponding improper integral is divergent:

$$\int_{x=1}^{\infty} \frac{1}{x}\, dx = \lim_{b\to\infty} \ln x \,|_1^b = \lim_{b\to\infty} \ln b = \infty.$$

Thus the harmonic series is also divergent, as suggested by the circumscribed rectangles representing the first five terms in the harmonic series as shown in the figure below.



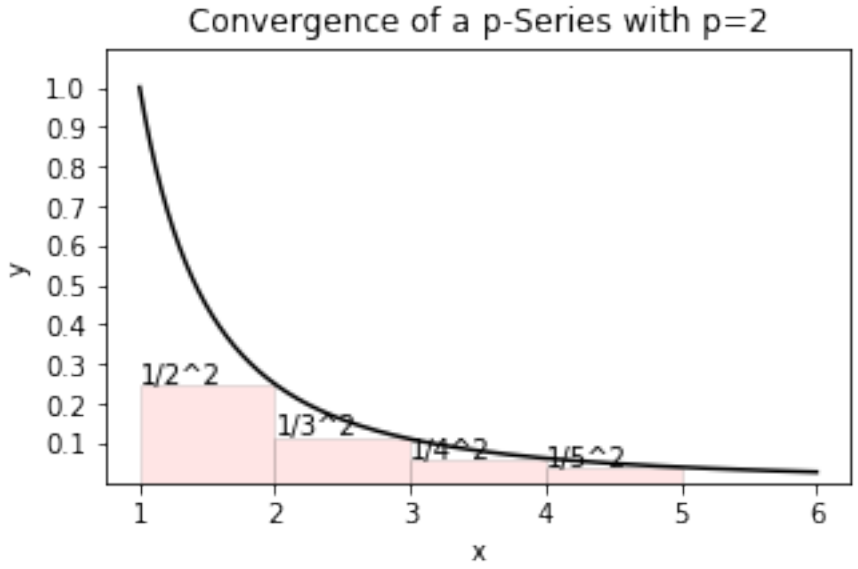
On the other hand, a p-series with $p = 2$ series must converge as shown by a convergent improper integral

$$\int_1^\infty \frac{1}{x^2}\,dx = \lim_{b\to\infty} -x^{-1}\,|_1^b = \lim_{b\to\infty} -\frac{1}{b} + 1 = 1.$$

in comparison with inscribed rectangles representing the tail of the series

$$\frac{1}{2^2} + \frac{1}{3^2} + ... < \int_1^\infty \frac{1}{x^2} = 1 \Rightarrow$$

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = 1 + \frac{1}{2^2} + \frac{1}{3^2} + ... < 1 + \int_1^\infty \frac{1}{x^2} = 2.$$



7.12.3 Income Streams

If P dollars are invested into an account with interest compounded continuously with annual interest rate $r = .05$ (i.e 5%), then after t years the amount P grows to $Pe^{rt} = Pe^{.05t}$.

Suppose $f(t) = 200 + 10t$ dollars are invested into an account with the same interest rate at times $t = 1, 2, 3$ (in years) and the total including interest is computed at $t = 3$.

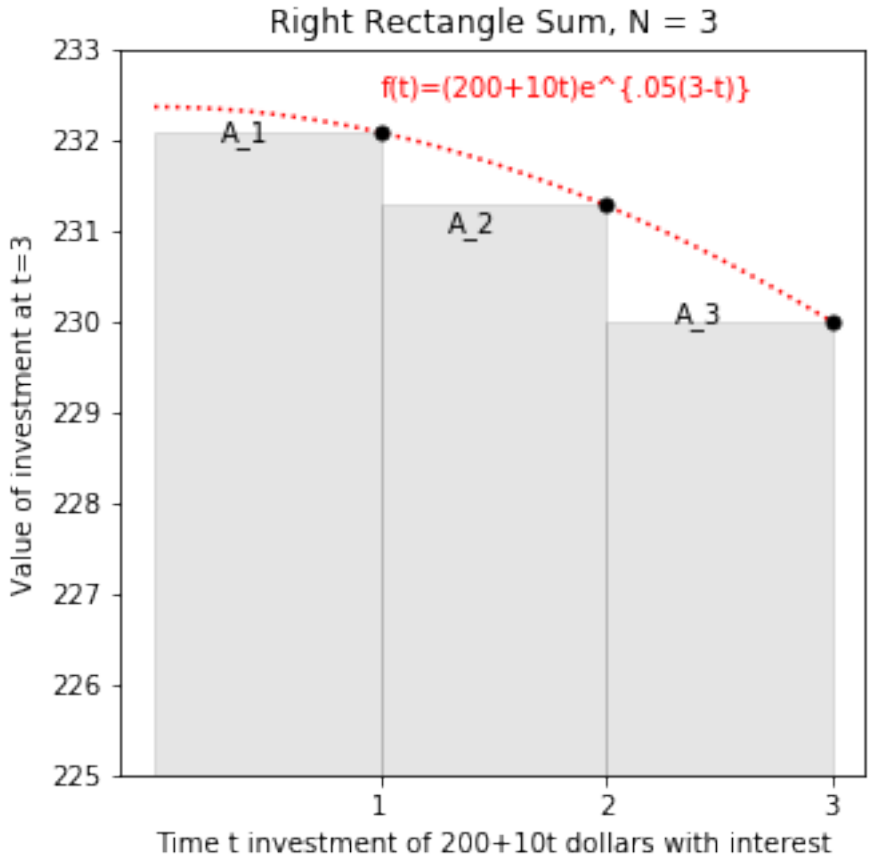
The amount \$210 invested at $t = 1$ grows to $A_1 = \$210e^{.05(3-1)} = 210e^{.05(2)}$.

The amount \$220 invested at $t = 2$ grows to $A_2 = \$220e^{.05(3-2)} = 220e^{.05(1)}$.

The amount \$230 invested at $t = 3$ grows to $A_3 = \$230e^{.05(3-3)} = 230e^{.05(0)} = 230$.

Note that

$$A_1 + A_2 + A_3 < \int_1^3 (200 + 10t)e^{.05(3-t)}.$$



7.12.4 Exercises

Exercises

1) Suppose $f(t) = 10,000 + 500t$ dollars are invested into an account (with the same interest rate) at times $t = 1, 2, 3$, and 4 (t in years). At time $t = 4$, find the total value of the investment with interest $A_1 + A_2 + A_3 + A_4$.
2) Show geometrically that the sum $10,500e^{.05(3)} + 11,000e^{.05(2)} + 11,500e^{.05} + 12,000$ is less than $I_4 = \int_0^4 (10,000 + 500t)e^{.05(4-t)}dt$.
3) Explain why

$$I_4 = \int_0^4 (10,000 + 500t)e^{.05(4-t)}dt = e^{.05(4)} \int_0^4 (10,000 + 500t)e^{-.05t}dt = e^{.05(4)}P_4.$$

4) Explain why an amount $P_4 = \int_0^4 (10,000+500t)e^{-.05t}dt$ invested at time $t = 0$ will allow withdrawals whose total value with interest is $A_1 + A_2 + A_3 + A_4$.
5) Extend the reasoning in exercises 1-4 to show that $P_N = \int_0^N (10,000 + 500t)e^{-.05t}dt$ deposited at $t = 0$ is sufficient to allow $f(t) = 10,000 + 500t$ to be withdrawn from the account at $t = 1, 2, ..., N$. Use integration by parts to find $P_\infty = \int_0^\infty (10,000 + 500t)e^{-.05t}dt$. (If this finite amount P_∞ is deposited at $t = 0$, then $f(t) = 10,000 + 500t$ can be withdrawn at $t = 1, 2, 3, ...$ ("in perpetuity".)

7.13 Ordinary Differential Equations and Exponential Growth/Decay

Simply put, a differential equation is an equation that includes at least one derivative of a function. For example,

$$\frac{dy}{dt} = y,$$

is a differential equation involving a function $y(t)$ and its first derivative $\frac{dy}{dt}$.

Differential equations are used widely across many areas of science, economics, and mathematics.

7.13.1 Initial Value Problems

An initial value problem (IVP) consists of a differential equation (DE) and an initial condition (IC). Here is an example:

$$(DE)\ \frac{dy}{dt} = \frac{1}{y},$$

$$(IC)\ y(0) = 1$$

7.13.2 Separation of Variables

A basic technique which can be used to solve simple IVP's is called *separation of variables*. Here is how we solve the above IVP using this technique:

1) bring to one side of the equation all the y's times dy, and to the other side of the equation all the $t's$ times dt:

$$y\,dy = \ dt.$$

2) Integrate both sides of the equation:

$$\int y\,dy = \int dt \Rightarrow \frac{y^2}{2} = t + C.$$

3) Use the initial condition (IC) $y(0) = 1$ (i.e. $y = 1$ when $t = 0$) to find the value of C:

$$\frac{1}{2} = 0 + C \Rightarrow C = \frac{1}{2}.$$

4) Substitute the value for C.

$$\frac{y^2}{2} = t + \frac{1}{2}.$$

5) Solve for y.

$$y^2 = 2t + 1.$$

$$y = \sqrt{2t + 1}.$$

Note that we use the positive square root to satisfy the IC $y(0) = 1$.

7.13.3 Exponential Growth

In some situations such as continuous compounding of interest, the rate of growth of a quantity is proportional to the size of the quantity:

$$\frac{dy}{dt} = ky.$$

The constant $k > 0$ is called the growth constant.

If at $t = 0$ the value of y is y_0 (that is we are given the IC: $y(0) = y_0$), we can use separation of variables to solve for $y(t)$:

$$\frac{dy}{y} = kdt$$

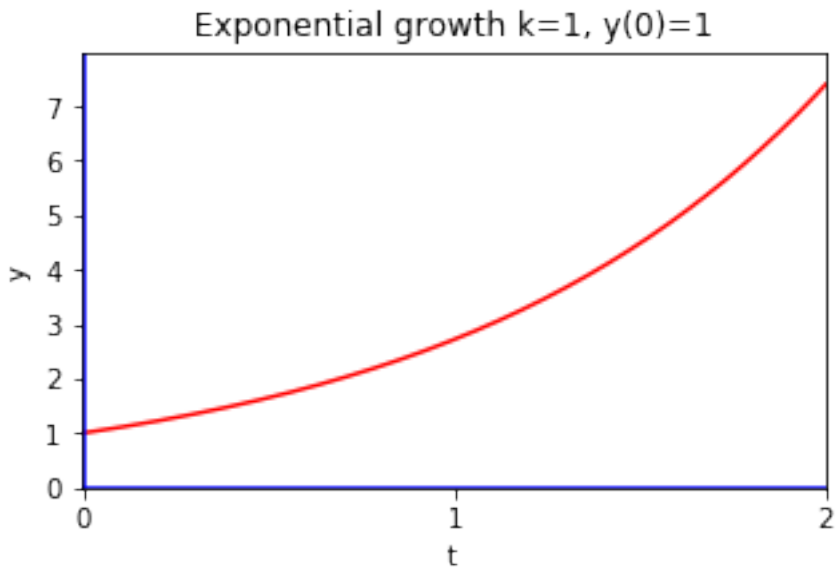
$$\int \frac{dy}{y} = k \int dt$$

$$\ln(y) = kt + C$$

$$y = e^{kt+c} = e^{kt}e^C = C_1 e^{kt}$$

$$y(0) = C_1 e^0 = C_1 = y_0.$$

Thus, our solution is $y(t) = y_0 e^t$.



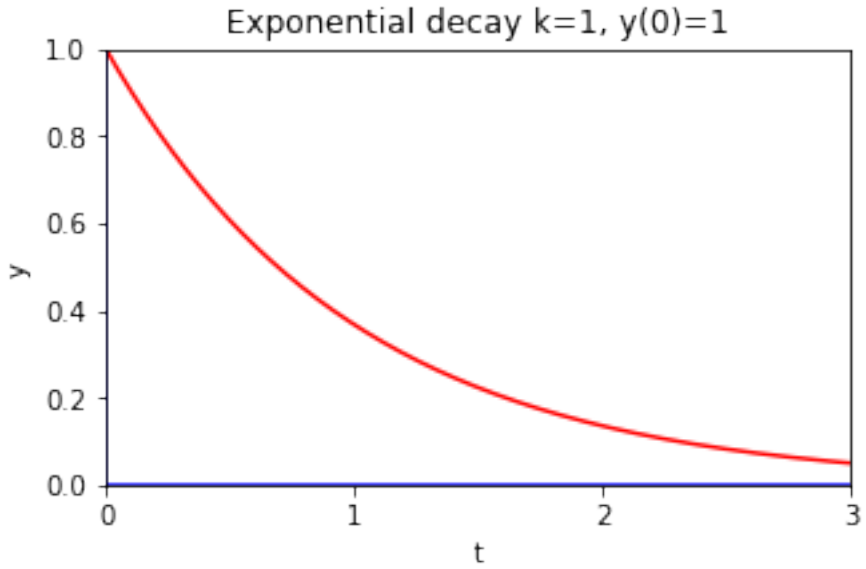
7.13.4 Exponential Decay

In some situations such as radioactive decay, the IVP is

$$\frac{dy}{dt} = -ky,\ y(0) = y_0.$$

Using separation variables in a similar way to solving the exponential growth equation leads to the solution

$$y(t) = y_0 e^{-kt}.$$



Half-Life

For radioactive decay problems, the time it takes for radioactive material to decay to half the original amount is called the *half-life*.

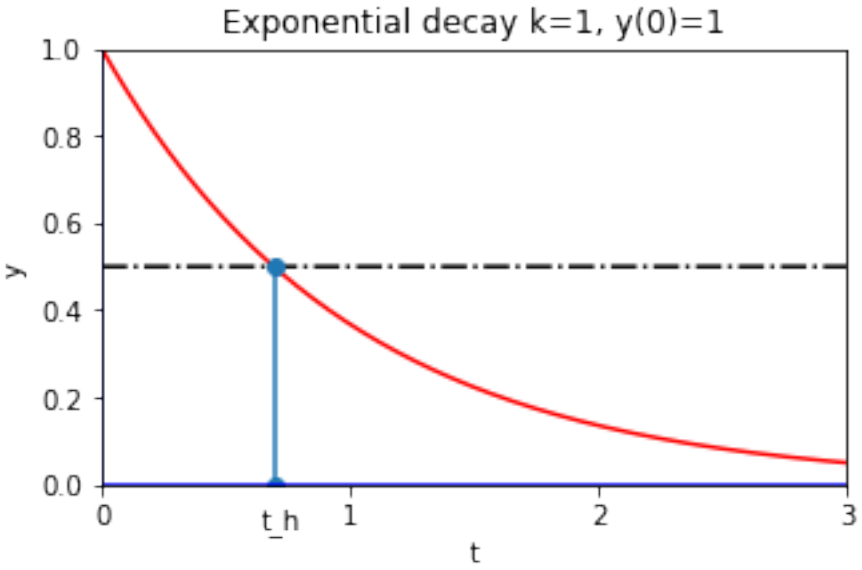
To find the half-life $t = t_h$ for the case $k = 1$, we solve as follows:

$$y(t_h) = y_0 e^{-t_h} = \frac{1}{2} y_0$$

$$e^{-t_h} = \frac{1}{2}$$

$$-t_h = \ln \frac{1}{2}$$

$$t_h = -\ln \frac{1}{2} = \ln 2 \approx .69.$$



7.13.5 Exercises

Exercises

1. Let $y(t)$ be the amount (in grams) of a pollutant at time t (in days) found in a certain water supply. Suppose after an industrial accident, $y(0) = 100$ and additional pollutant enters the water supply at a constant rate of $c_1 = 5$ gm/day. Emergency water purification measures are used to reduce the daily amount of pollutant at a rate of 1% of the existing amount of pollution.

a) What IVP is used to model this situation?

Hint: The rate of change of pollutant $\frac{dy}{dt}$ is equal to rate in - rate out.

b) Use separation of variables to solve the IVP.

c) Suppose the water supply becomes unsafe if the pollution level reaches 400 gm. After how many days will the water supply become unsafe?

d) Make a graph of $y(t)$.

2. Prove the Rule of 72: If money is deposited into an account with an annual interest rate r (expressed as a percentage) compounded continuously, then it will take roughly $72/r$ years for the money to double.

3. For exponential decay, find a formula relating the decay constant k and the half-life t_h.

4. The exponential growth model predicts that $y \to \infty$ as $t \to \infty$. This may not be realistic. For example, if $y(t)$ represents the population at time t, there is a limit to how big $y(t)$ can grow.

The logistic model includes a cap M on how big the population can grow, and is given by the following IVP:

$$\frac{dy}{dt} = ky(M-y),\ y(0) = y_0.$$

Use separation of variables to solve the logistic growth model for the case $k = 1$, $M = 100$, $y_0 = 5$.

Hint:

$$\frac{1}{y(M-y)} = \frac{1}{M}(\frac{1}{y} + \frac{1}{M-y}).$$

7.14 Solutions to Exercises

7.14.1 Linear Systems and the Two-Commodity Model

1. Let x be the units of necessities and y the units of luxuries. Then $30x + 100y = 4000$ and $y = \frac{1}{10}x$. This implies that $40x = 4000$ so $x = 100$ and $y = 10$.

2. A 10% decrease in budget gives $30x + 100y = 3600$ with $y = \frac{1}{10}x$. The equilibrium bundle is $x = 90$, $y = 9$. A 10% increase in unit prices gives $33x + 110y = 4000$ with $y = \frac{1}{10}x$ so $x = 90.9$ and $y = 9.09$. So a decreased budget results in a greater loss of buying power.

3. A 10% increase in budget gives $30x + 100y = 4400$, $y = \frac{1}{10}x$. The equilibrium bundle is $x = 110, y = 11$. A 10% decrease in prices gives $27x + 9y = 4000$, $y = \frac{1}{10}x$. The equilibrium bundle is $x = 111.1, y = 11.1$. So the decrease in prices has the greater gain in buying power.

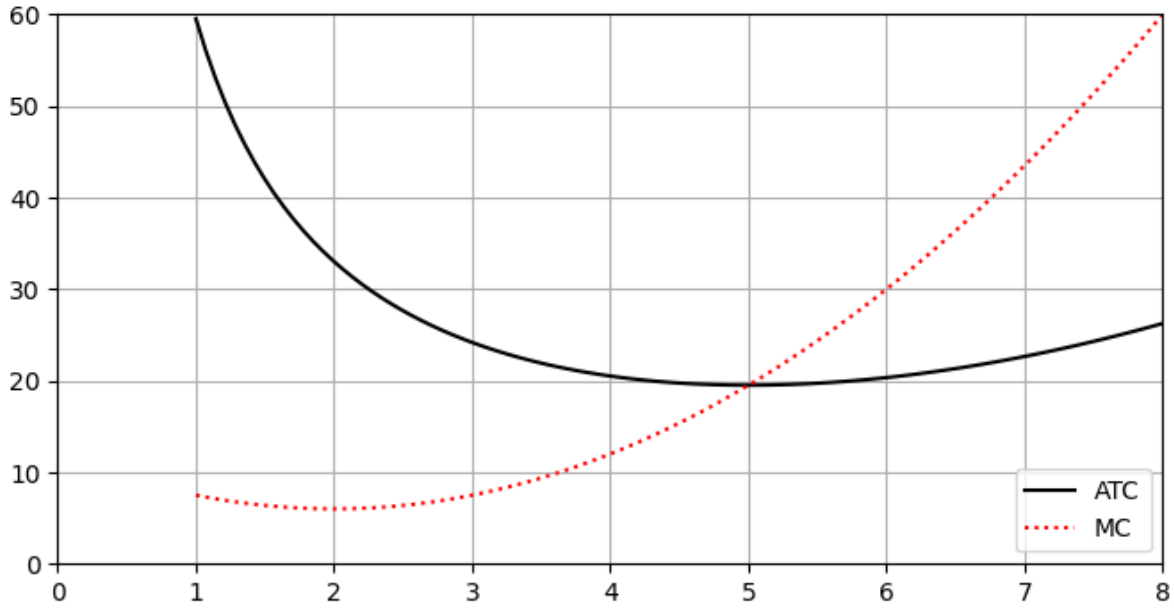
4. The solid black line is the original budget. The other lines in order of increasing buying power are 10% decrease of income, 10% increase in prices, 10% increase in income, and 10% decrease in prices.

7.14.2 Marginal and Average Cost

1.

$$ATC = \frac{TC}{Q} = .5Q^2 - 3Q + 12 + 50/Q$$

$$MC = TC'(Q) = 1.5Q^2 - 6Q + 12$$



From the graph it appears ATC=MC at $Q = 5$. We can check this:

$$ATC(5) = .5(5^2) - 3(5) + 12 + \frac{50}{5} = 19.5.$$

$$MC(5) = 1.5(5^2) - 6(5) + 12 = 19.5.$$

Finally, note that

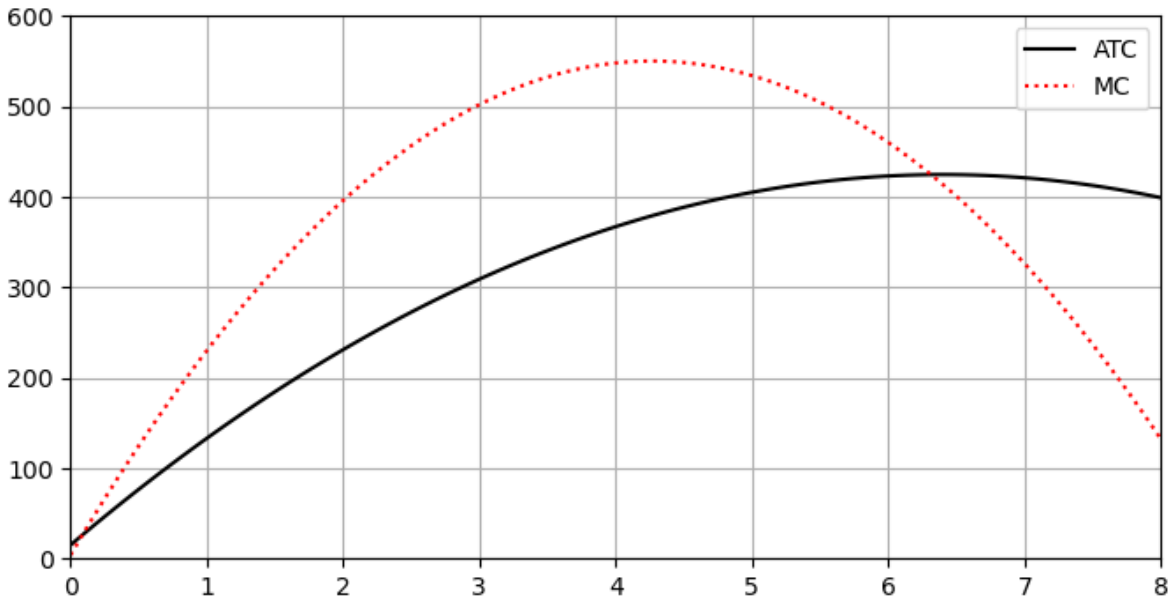
$$ATC'(Q) = Q - 3 - \frac{50}{Q^2}.$$

At $Q = 5$, $ATC'(5) = 5 - 3 - \frac{50}{5^2} = 0$.

2.

$$ATC = \frac{TC}{Q} = -10Q^2 + 128Q + 3.$$

$$MC = TC'(Q) = -30Q^2 + 256Q + 3.$$



The graph does not show clearly the value of Q at the intersection. Note that $ATC'(Q) = -20Q + 128 = 0$ when $Q = 6.4$.

We can check that $ATC(6.4) = MC(6.4) = 412.6$.

3.

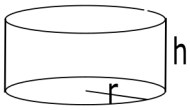
$ATC = TC/Q$, so by the quotient rule, $ATC'(Q) = 0$ when

$$\frac{Q \cdot MC - TC}{Q^2} = 0.$$

This occurs when $Q \cdot MC - TC = 0$, or $MC = \frac{TC}{Q} = ATC$.

7.14.3 Optimization and Object Design

1.

a) 

b) $h = \frac{250}{\pi x^2}$.

c) $f(x) = 2\pi x^2 + 500x^{-1}$.

d) $f'(x) = 4\pi x - 500x^{-2}$

e) $x = (125/\pi)^{1/3}$

f) $h = 2(125/\pi)^{1/3} = 2x$.

g) The height of the can and its diameter are equal. The can is 'square-shaped'.

h)

$$f'(x) = 4\pi x - 500x^{-2}$$

$$f''(x) = 4\pi + \frac{1000}{x^3}$$

When $x = (125/\pi)^{1/3}$. $f''(x) > 0$, hence there is a minimum value for f.

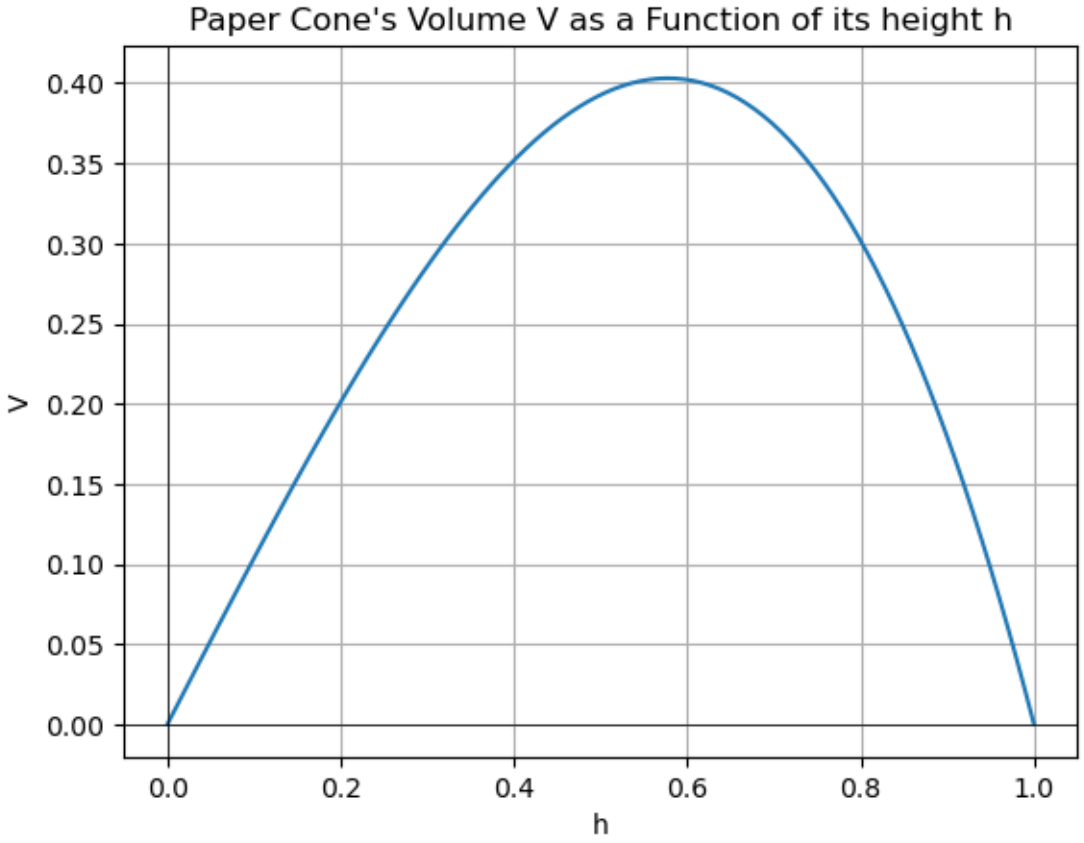
2. a) Volume $V(h) = \frac{\pi}{3}(r^2 h - h^3)$

$V'(h) = \frac{\pi}{3}(r^2 - 3h^2) = 0$ when $h = \pm\frac{r}{\sqrt{3}}$. ($h$ must be positive)

$V(\frac{r}{\sqrt{3}}) = \frac{2\pi r^3}{9\sqrt{3}}$.

b) $V''(h) = \frac{\pi}{3}(-6h) < 0$ when $h = \frac{r}{\sqrt{3}}$ so the volume is maximized.

c) The graph and code to make it are shown below:



7.14.4 Optimization and Cobbs-Douglas Production

1.

$$10L + 20K + 20 = 3040 \Rightarrow 20K = 3020 - 10L \Rightarrow K = 151 - .5L.$$

$$Q = 2L^{.4}(151 - .5L)^{.6}$$

$$\frac{dQ}{dL} = 2[.4L^{-.6}(151 - .5L)^{.6} + L^{.4}(.6)(151 - .5L)^{-.4}(-.5)] =$$

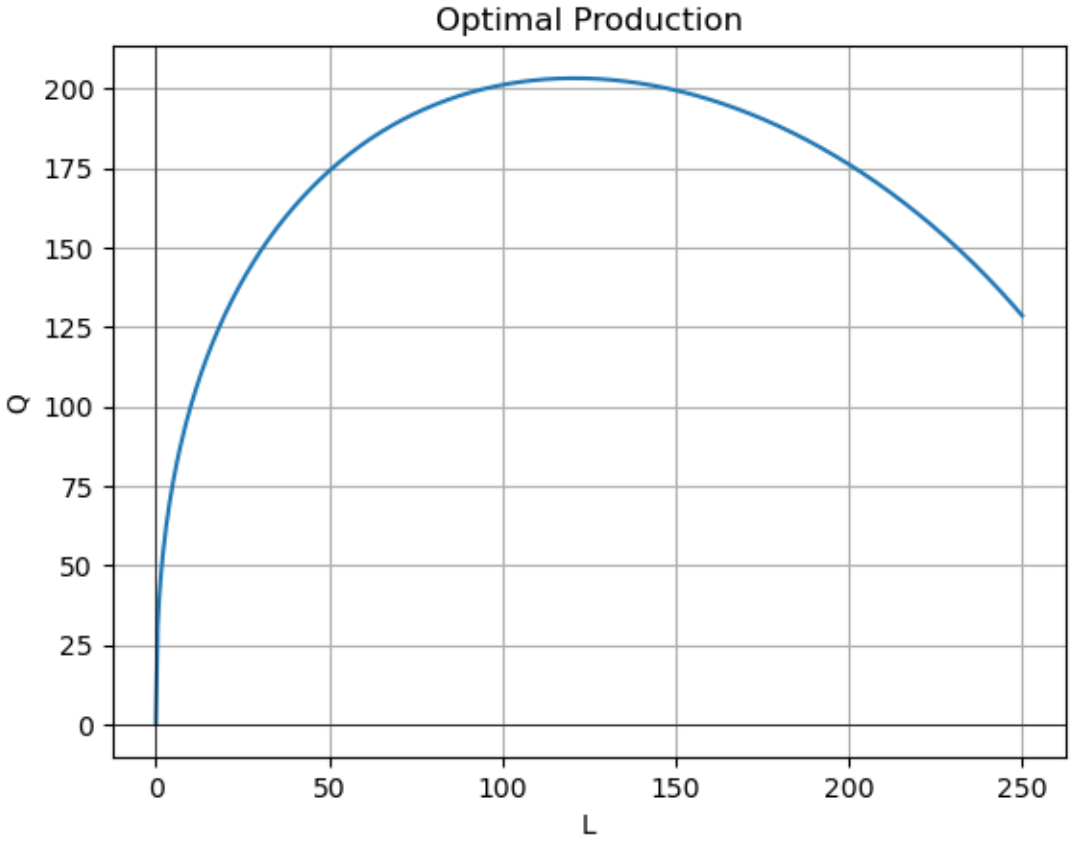
$$\frac{.8(151 - .5L)^{.6}}{L^{.6}} - \frac{.6L^{.4}}{(151 - .5L)^{.4}} = 0 \Rightarrow$$

$$.8(151 - .5L) = .6L.$$

Solving the last equation for L gives $L = 120.8$.

$$K = 151 - .5L = 151 - .5(120.8) = 90.6$$

The maximum production is $Q = 2(120.8)^{.4}(90.6)^{.6} = 203.3$.



2.

$$130 = 2L^{.4}K^{.6} \Rightarrow \frac{130}{2L^{.4}} = K^{.6} \Rightarrow K = (\frac{65}{L^{.4}})^{5/3} = 65^{5/3}L^{-2/3}$$

$$TC = 10L + 20(65^{5/3}L^{-2/3}) + 20$$

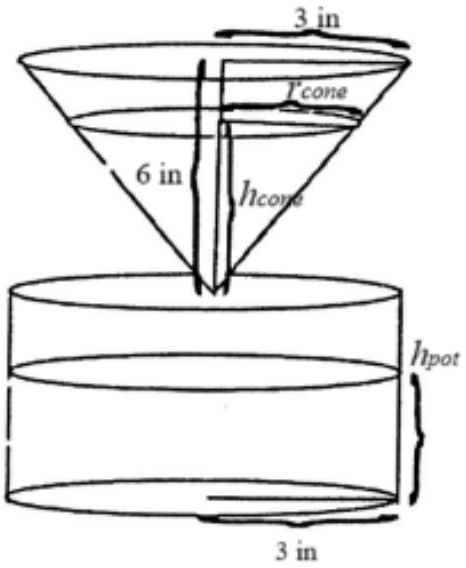
$$TC' = 10 - \frac{40}{3}65^{5/3}L^{-5/3} = 0 \Rightarrow L = 77.246$$

Note that $TC'' = \frac{200}{9}65^{5/3}L^{-7/3} > 0$ so there is a minimum at $L = 77.25$.

$K = 65^{5/3}(77.246)^{-2/3} \approx 57.9346$.

The minimum total cost is $10(77.246) + 20(57.9346) + 20 \approx 1951.15$.

7.14.5 Related Rates and Volumes



1.

2. $\frac{r_{cone}}{h_{cone}} = 1/2 \Rightarrow r_{cone} = \frac{1}{2}h_{cone}$

3. $v_{cone} = \frac{1}{3}\pi r_{cone}^2 h_{cone} = \frac{\pi}{3}\frac{h_{cone}^2}{4}h_{cone} = \frac{\pi}{12}h_{cone}^3$

4. $\frac{d}{dt}[V_{cone}] = \frac{d}{dt}[\frac{\pi}{12}h_{cone}^3] = \frac{\pi}{4}h_{cone}^2\frac{dh_{cone}}{dt}$

5. $v_{pot} = \pi r^2 h_{pot} = \pi 3^2 h_{pot} = 9\pi h_{pot}$

6. $\frac{d}{dt}[V_{pot}] = \frac{d}{dt}[9\pi h_{pot}] = 9\pi\frac{dh_{pot}}{dt}$

7. $\frac{d}{dt}[V_{cone}] = -\frac{d}{dt}[V_{pot}]$

8.

$$\frac{\pi}{4}h_{cone}^2\frac{dh_{cone}}{dt} = -9\pi\frac{dh_{pot}}{dt} \Rightarrow$$

$$\frac{\pi}{4}(5)^2\frac{dh_{cone}}{dt} = -9\pi(1) \Rightarrow$$

$$\frac{25\pi}{4}\frac{dh_{cone}}{dt} = -9\pi \Rightarrow$$

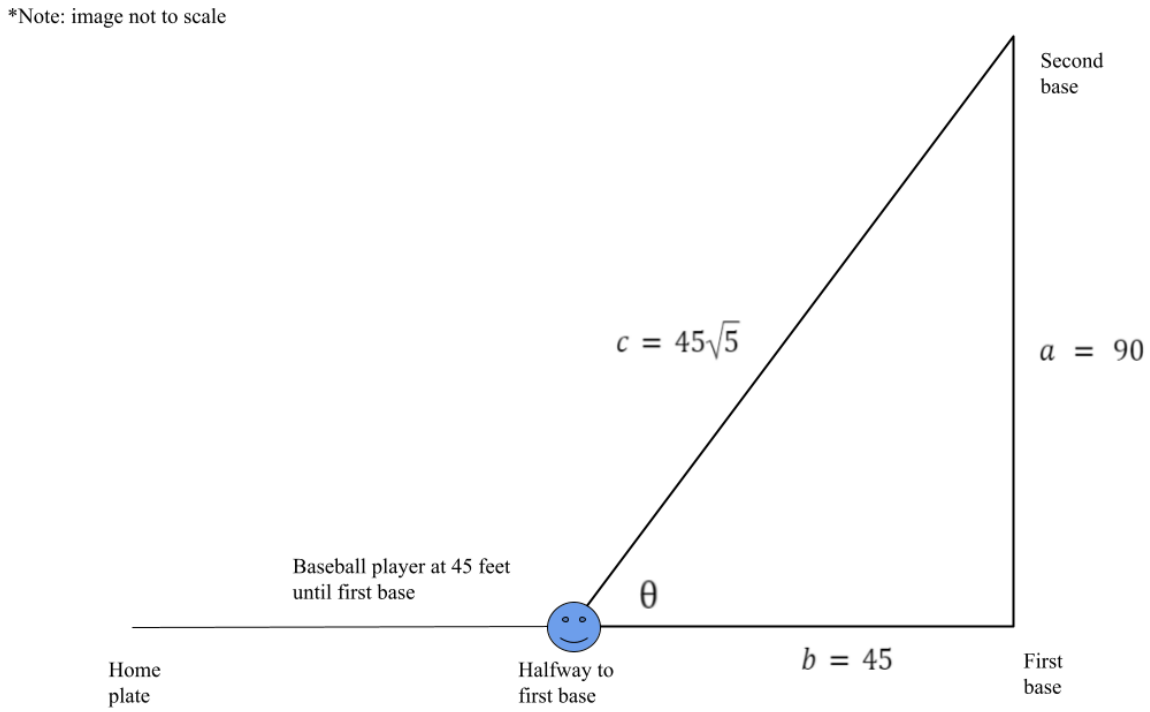
$$\frac{dh_{cone}}{dt} = \frac{-36\pi}{25\pi} \Rightarrow$$

$$\frac{dh_{cone}}{dt} = \frac{-36}{25}$$

9. We assume the coffee flows out of the filter at a continuous rate.

7.14.6 Related Rates with Trig Functions and Football

1. Using the Pythagorean theorem, we can create and label a triangle to represent the problem:



We are given $\frac{db}{dt} = -20$, but want to know what $\frac{dc}{dt}$ is when $b = 45$. Since the problem can be represented as a right triangle, we can further use the Pythagorean theorem and derivatives to solve for $\frac{dc}{dt}$.

$$c^2 = a^2 + b^2 \Rightarrow$$

$$c^2 = 8100 + b^2 \Rightarrow$$

$$2c\frac{dc}{dt} = 2b\frac{db}{dt} \Rightarrow$$

$$45\sqrt{5}\frac{dc}{dt} = (90)(-20) = -180 \Rightarrow$$

$$\frac{dc}{dt} = -\frac{4\sqrt{5}}{5} \approx -1.789$$

We are given $\frac{db}{dt} = -20$, but want to know what $\frac{d\theta}{dt}$ is when $b = 45$. Using the triangle and trigonometric functions, we can solve for $\frac{dc}{dt}$.

$$\tan\theta = \frac{9}{b} \Rightarrow$$

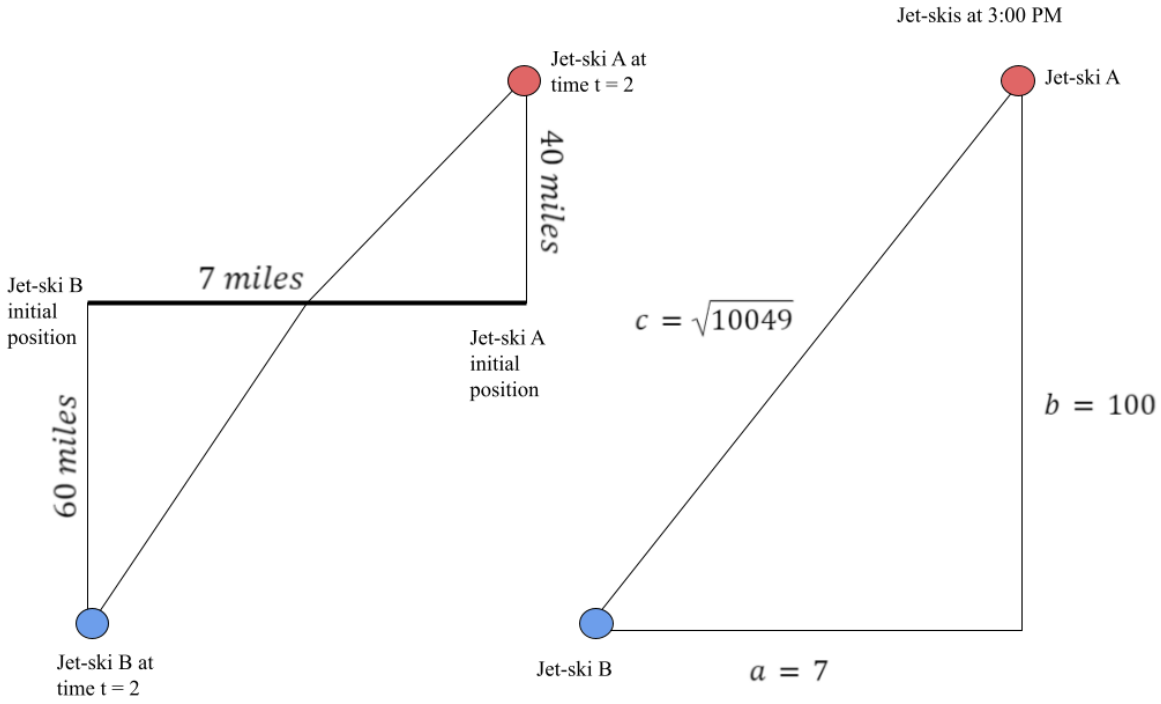
$$\sec\theta^2\frac{d\theta}{dt} = -\frac{90}{b^2} \Rightarrow$$

$$\frac{d\theta}{dt} = -\frac{90}{b^2}\cos\theta^2 \Rightarrow$$

$$\frac{d\theta}{dt} = -\frac{90}{2025}(\frac{45}{45\sqrt{5}})^2 = -\frac{90}{2025}\frac{1}{5} = -\frac{2}{225}$$

2. Using algebra to find the sidelength and the Pythagorean theorem, we can create and label a triangle to represent the problem:

*Note: image not to scale



We are given $\frac{db}{dt} = 50$, but want to know what $\frac{dc}{dt}$ is when $b = 100$. Since the problem can be represented as a right triangle, we can further use the Pythagorean theorem and derivatives to solve for $\frac{dc}{dt}$.

$$c^2 = a^2 + b^2 \Rightarrow$$

$$c^2 = 49 + b^2 \Rightarrow$$

$$2c\frac{dc}{dt} = 2b\frac{db}{dt} \Rightarrow$$

$$\sqrt{10049}\frac{dc}{dt} = (100)(50) = 5000 \Rightarrow$$

$$\frac{dc}{dt} = \frac{5000\sqrt{10049}}{10049} \approx 49.878$$

7.14.7 Probability Distributions and Drive Thrus

1. a) $\int_0^2 e^{-t}\,dt = -e^{-t}\,|_0^2 = -e^{-2} - -e^0 = 1 - e^{-2}$

b) $\int_0^\infty ke^{-t}\,dt = -e^{-t}\,|_0^\infty = \lim_{t\to\infty} -e^{-t} + e^0 - -e^0 = 0 + 1 = 1$

c)

$$\int_0^1 ke^{-kt}\,dt = 0.5 \Rightarrow$$

$$-e^{-kt}\,|_0^1 = 0.5 \Rightarrow$$

$$-e^{-k(1)} - -e^{-k(0)} = -e^{-k} + 1 = 0.5 \Rightarrow$$

$$e^{-k} = 0.5 \Rightarrow$$

$$-k = \ln 0.5 \Rightarrow$$

$$k = -\ln 0.5 \approx 0.693$$

d)

$$F(x) = \int_0^x -ln0.5e^{\ln 0.5t}\, dt =$$

$$\int_0^x -ln0.50.5^t\, dt =$$

$$-0.5^t \mid_0^x =$$

$$-0.5^x - -0.5^0 =$$

$$-0.5^x + 1 =$$

$$1 - 0.5^x$$

e) $F(3) = 1 - 0.5^3 = 0.875$

2. a)

$$f(x) = \frac{1}{30}(5 \leq x \leq 35)$$

$$F(t) = prob(5 \leq x \leq t) = \int_5^t 1/30\, dt = \frac{1}{30}x \mid_5^t = \frac{t-5}{30}$$

$$r = \frac{P-5}{30} \Rightarrow$$

$$P = 30r + 5$$

b)

$$f(x) = \frac{1}{15}(5 \leq x \leq 20)$$

$$F(t) = prob(5 \leq x \leq t) = \int_5^t 1/15\, dt = \frac{1}{15}x \mid_5^t = \frac{t-5}{15}$$

$$r = \frac{P-5}{15} \Rightarrow$$

$$P = 15r + 5$$

3.

$$g(x) = \frac{1}{60}(30 \leq x \leq 90)$$

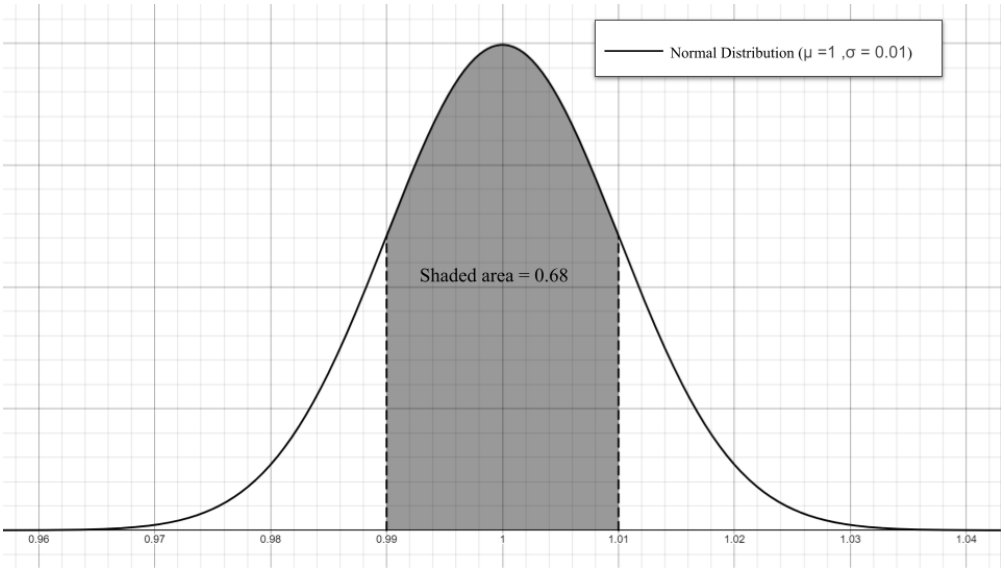
$$g(t) = prob(30 \leq x \leq t) = \int_{30}^t 1/60\, dt = \frac{1}{60}x \mid_{30}^t = \frac{t-30}{60}$$

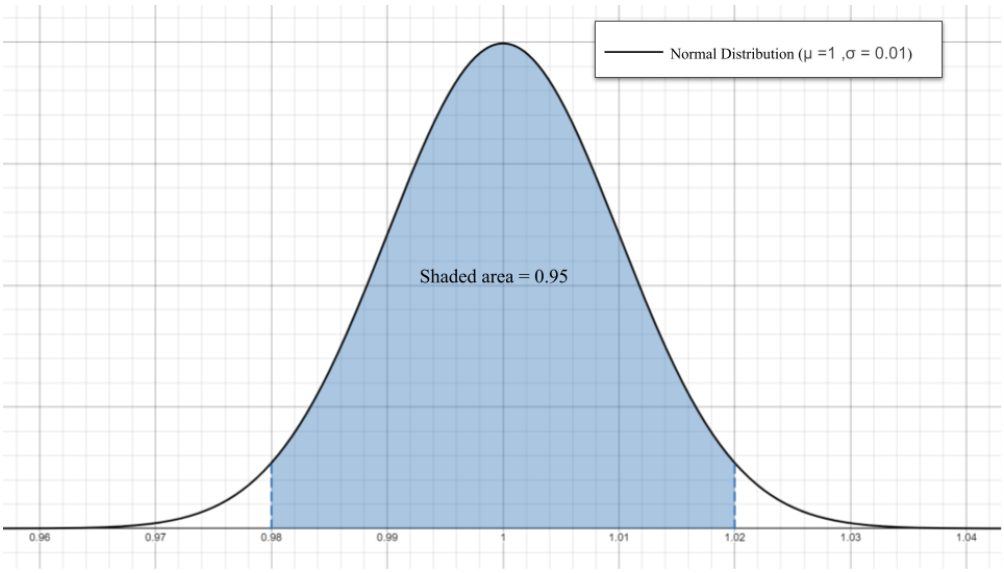
$$r = \frac{F-30}{60} \Rightarrow$$

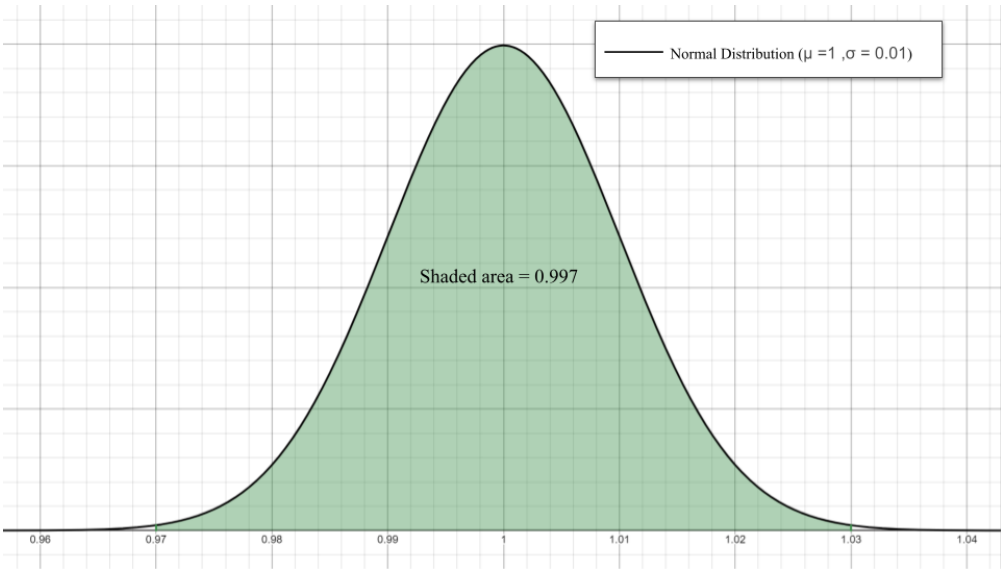
$$F = 60r + 30$$

7.14.8 Normal Distribution and Process Control

1.





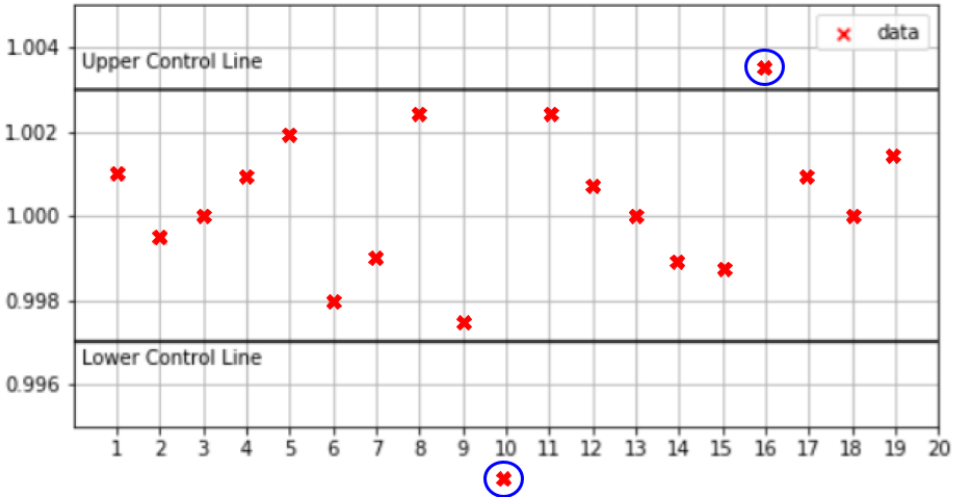


2. a) $100 \cdot 0.997 = 99.7$

b) $100 \cdot (0.003/2) = 0.15$

c) $99.7 + 0.15 = 99.85$

3. a)



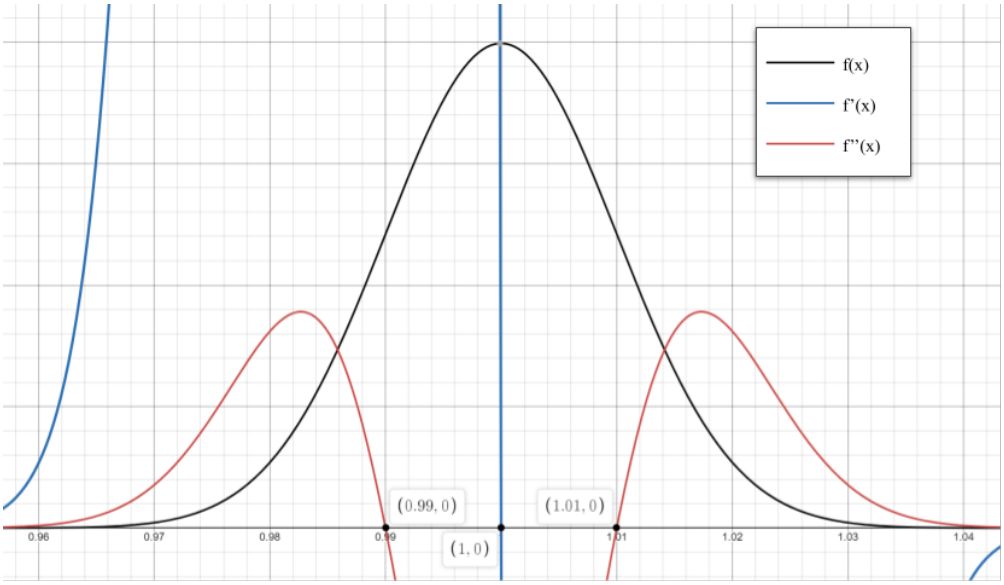
b) Yes, 0.9825 and 1.0035

4.

$$f(x) = \frac{1}{1 \cdot \sqrt{2\pi}} e^{-\frac{(x-1)^2}{2 \cdot 0.01^2}}$$

$$f'(x) = \frac{1}{1 \cdot \sqrt{2\pi}} e^{-\frac{(x-1)^2}{2 \cdot 0.01^2}} [\frac{-(x-1)}{0.01^2}]$$

$$f''(x) = \frac{1}{1 \cdot \sqrt{2\pi}} e^{-\frac{(x-1)^2}{2 \cdot 0.01^2}} [\frac{(x-1)^2}{0.01^4} - \frac{1}{0.01^2}]$$

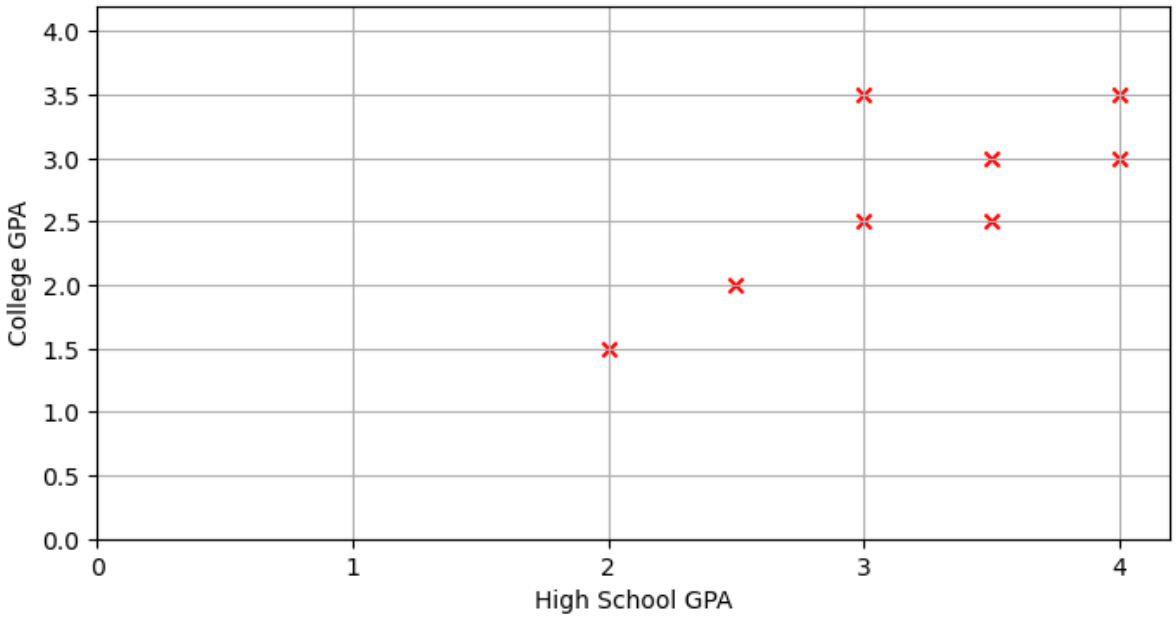


From the first derivative test, we can see that there is an absolute maximum at $x = 1$. Generally, the absolute minimum is found at $x = \mu$ for normal distribution functions.

From the second derivative test, we can see that there are inflection points at $(0.99, 24.197)$ and $(1.01, 24.197)$. Generally, the inflection points are found at $x = \mu \pm \sigma$ for normal distribution functions.

7.14.9 Partial Derivatives and OLS Regression

1. The plot and code to make it are shown below:



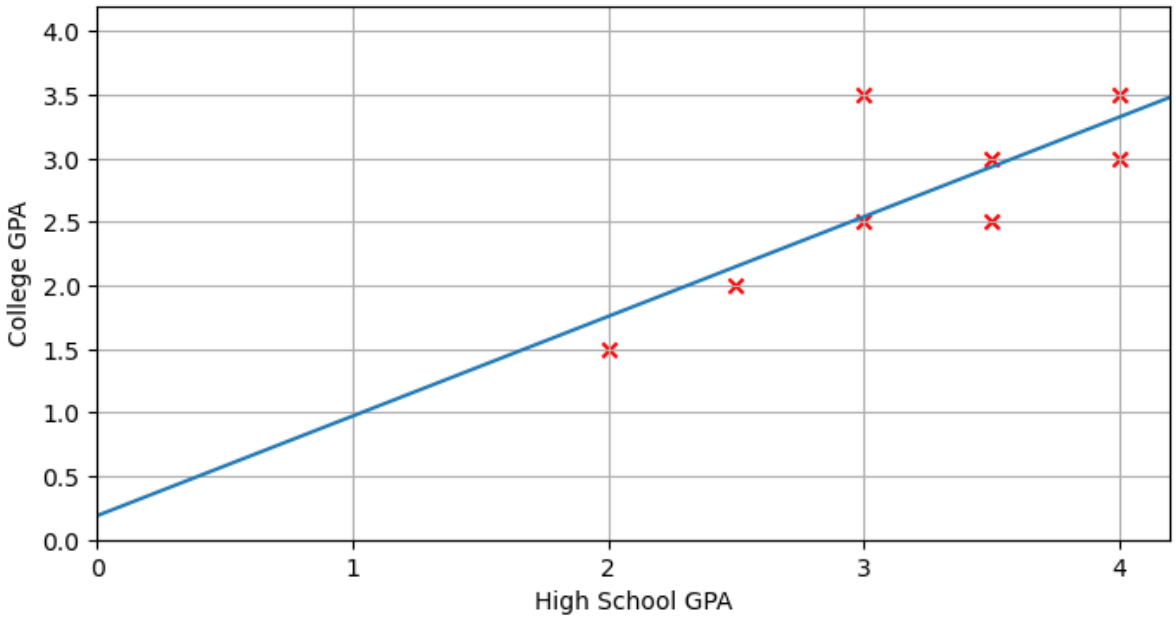
2.

$S = (2m + b - 1.5)^2 + (2.5m + b - 2)^2 + (3m + b - 2.5)^2 + (3m + b - 3.5)^2 + (3.5m + b - 2.5)^2 + (3.5m + b - 3)^2 + (4m + b$

$\frac{\partial S}{\partial m} = 4(2m + b - 1.5) + 5(2.5m + b - 2) + 6(3m + b - 2.5) + 6(3m + b - 3.5) + 7(3.5m + b - 2.5) + 7(3.5m + b - 3) + 8(4m$

$\frac{\partial S}{\partial b} = 2(2m + b - 1.5) + 2(2.5m + b - 2) + 2(3m + b - 2.5) + 2(3m + b - 3.5) + 2(3.5m + b - 2.5) + 2(3.5m + b - 3) + 2(4m$

Solving the system gives $m \approx .784$, $b \approx .189$.



Solution to exercise 3

$$\frac{\partial^2 S}{\partial m^2} = 169.5$$

$$\frac{\partial^2 S}{\partial b^2} = 16$$

$$\frac{\partial^2 S}{\partial m \partial b} = 51$$

$$D = (169.5)(16) - 51^2 = 111 > 0$$

Since $D > 0$ and $S_{mm} > 0$, S is minimized.

Solution to Exercise 4

GPA			
High School GPA:	2.0	3.0	4.0
College GPA:	1.76	2.54	3.3

7.14.10 Area between curves and the GINI index

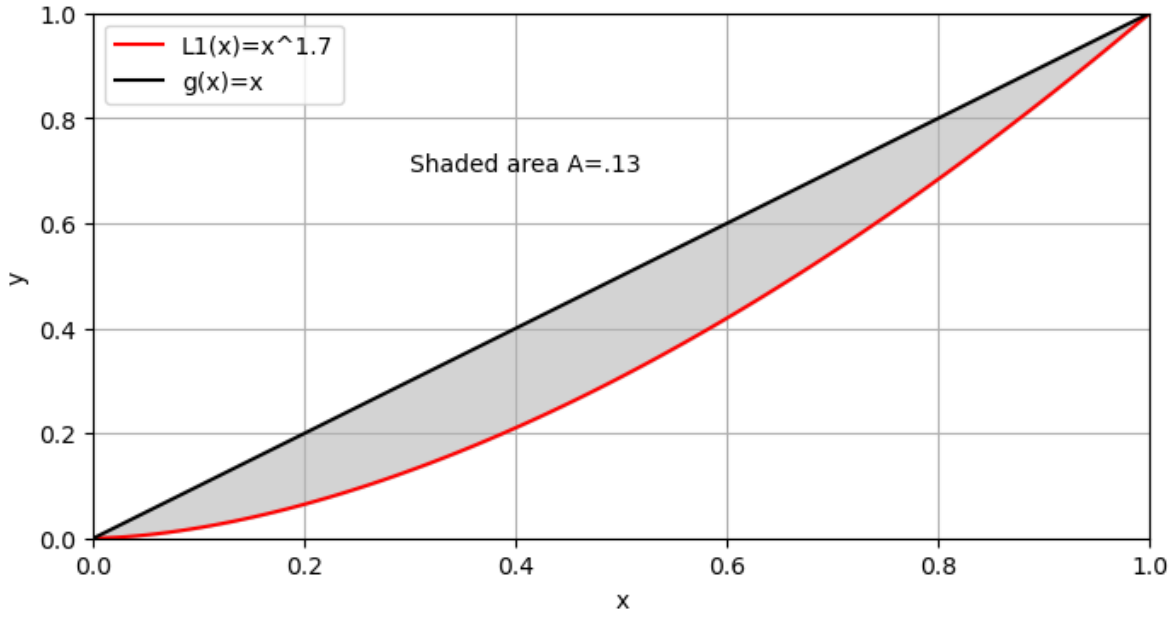
1.

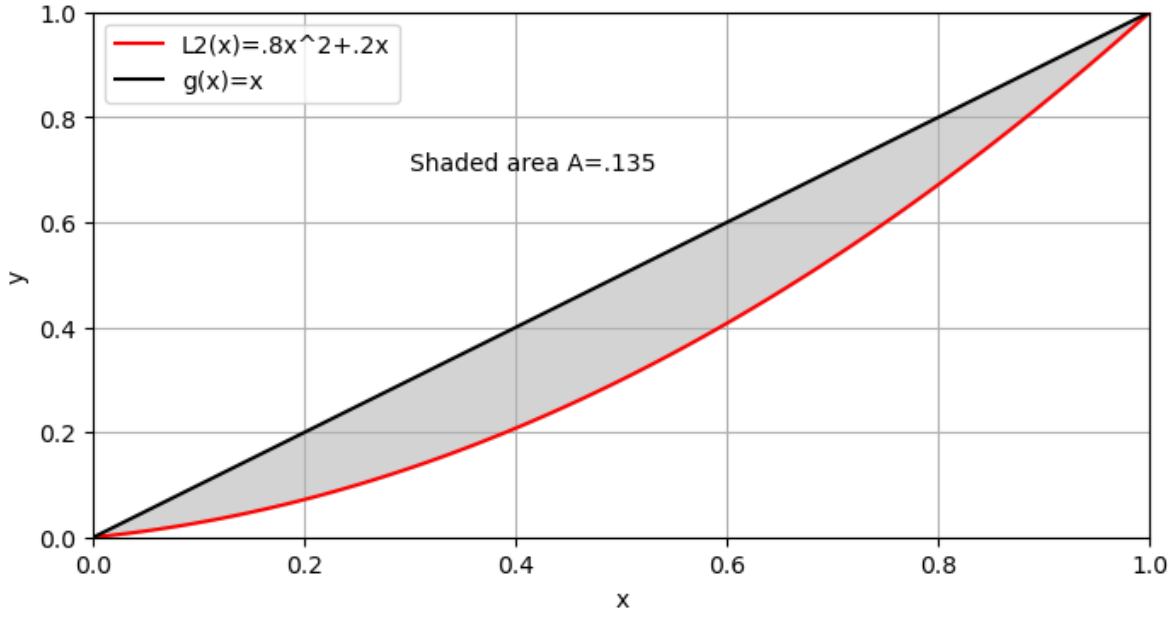
$$GI_1 = 2\int_0^1 (x - x^{1.7})\,dx = 2(\frac{x^2}{2} - \frac{x^{2.7}}{2.7})\,|_0^1 = 2(\frac{1}{2} - \frac{1}{2.7}) \approx .259.$$

$$GI_2 = 2\int_0^1 (x - (.8x^2 + .2x))\,dx = 2\int_0^1 (.8x - .8x^2)\,dx 2(.8)(\frac{x^2}{2} - \frac{x^3}{3})\,|_0^1 = 1.6(\frac{1}{2} - \frac{1}{3}) \approx .27.$$

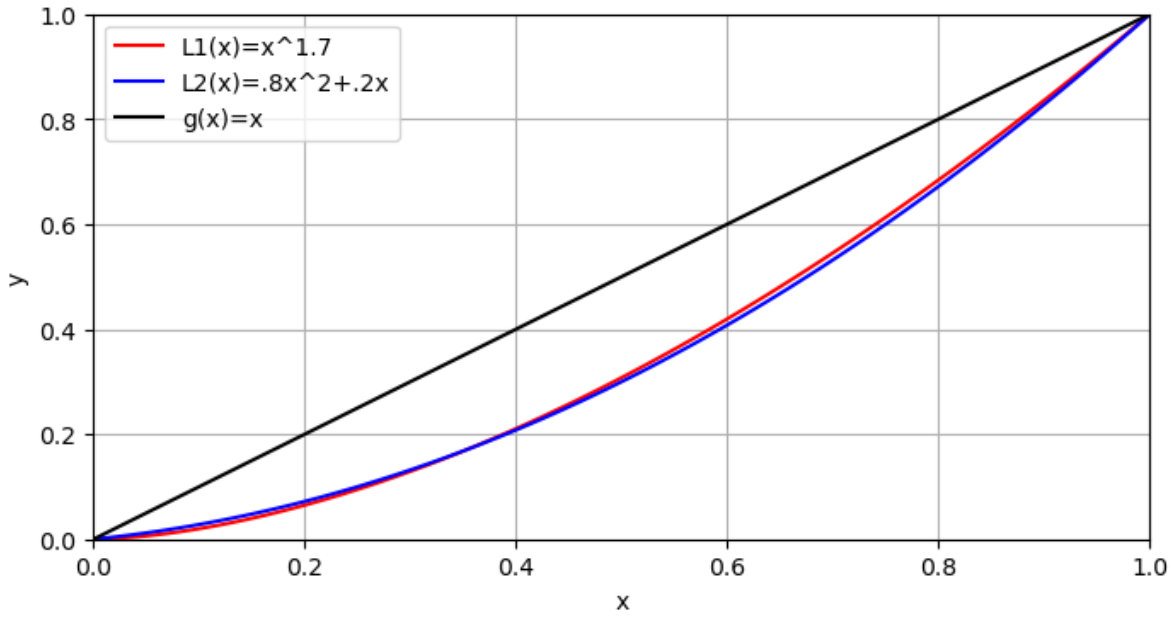
Profession 1 has a slightly more even salary distribution.

2.





One further plot shows the two Lorentz curves nearly coincide.

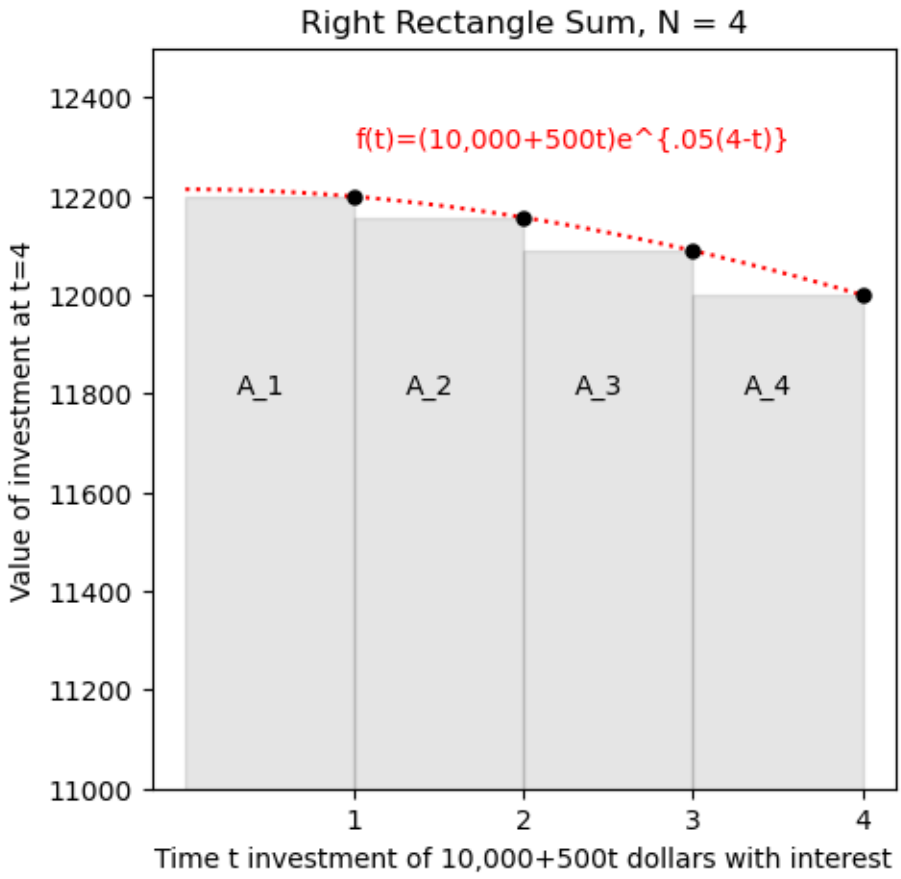


7.14.11 Integral Test and Income Streams

1. The total at $t = 4$ will be

$$10,500e^{.05(3)} + 11,000e^{.05(2)} + 11,500e^{.05} + 12,000 \approx 48,446.$$

2.



3. Note that

$$e^{.05(4-t)} = e^{.05(4)}e^{.05(-t)}.$$

4. Note that if we invest P_4 at $t = 0$, this amount will grow by the end of year 4 to an amount $e^{.05(4)}P_4 \approx 48,561.10$. The latter amount is the area under the curve $e^{.05(4-t)}$ on the interval 0, which by problem 2) is greater than $A_1+A_2+A_3+A_4$.

5.

$$P_\infty = \int_0^\infty (10,000 + 500t)e^{-.05t} =$$

$$(10,000 + 500t)(-20e^{-.05t}) \mid_0^\infty +10,000(-20)e^{-.05t} \mid_0^\infty =$$

$$200,000 + 200,000 = 400,000.$$

7.14.12 Ordinary Differential Equations and Exponential Growth/Decay

1. a) $\frac{dy}{dt} = 5 - 0.01y$

b)

$$\frac{dy}{dt} = 5 - 0.01y \Rightarrow$$

$$\int \frac{1}{5 - 0.01y}\, dt = \int dt \Rightarrow$$

$$-100 \ln |5 - 0.01y| = t + c \Rightarrow$$

$$\ln |5 - 0.01y| = \frac{t + c}{-100} \Rightarrow$$

$$5 - 0.01y = Ce^{\frac{t}{-100}} \Rightarrow$$

$$5 - Ce^{\frac{t}{-100}} = 0.01y \Rightarrow$$

$$y = 500 - Ce^{\frac{t}{-100}}$$

Plug in $y(0) = 100$ to find the value of C.

$$(100) = 500 - Ce^{\frac{(0)}{-100}} \Rightarrow$$

$$100 = 500 - C \Rightarrow$$

$$C = 400$$

The final equation is $y = 500 - 400e^{\frac{t}{-100}}$

c)

$$400 = 500 - 400e^{-\frac{t}{100}} \Rightarrow$$

$$100 = 400e^{-\frac{t}{100}} \Rightarrow$$

$$\frac{1}{4} = e^{-\frac{t}{100}} \Rightarrow$$

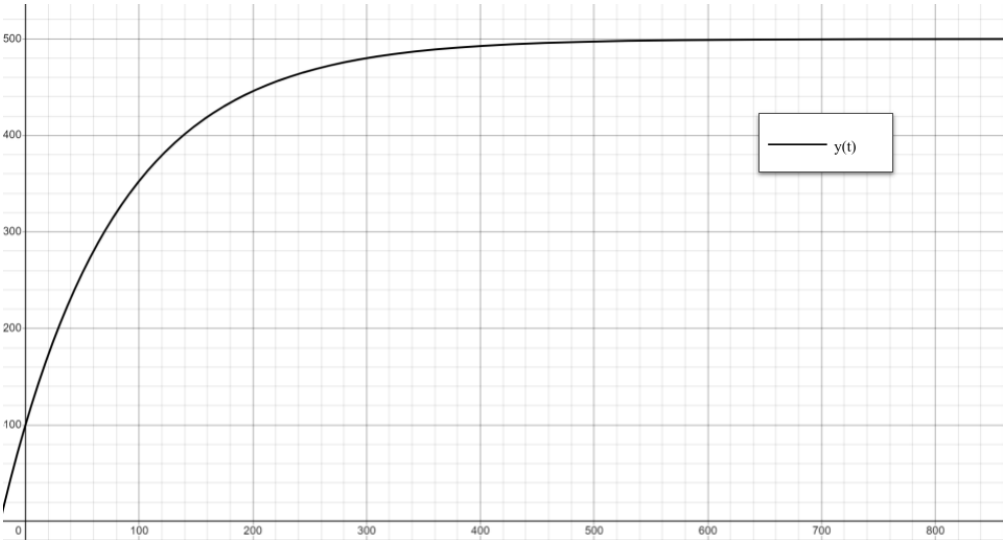
$$\ln \frac{1}{4} = \ln e^{-\frac{t}{100}} \Rightarrow$$

$$-2 \ln 2 = -\frac{t}{100} \Rightarrow$$

$$t = 200 \ln 2 \approx 138.629$$

The pollution level will become unsafe during the 138th day.

d)



2. We can prove the rule of 72 using the formula for compound interest, $A = P(1 + \frac{r}{100})^t$, where A is the amount accumulated, P is the principal amount, r is the interest rate, and t is the time (in years) the money has been invested. We are trying to prove that $t = \frac{72}{r}$ when $A = 2P$.

$$2P = P(1 + \frac{r}{100})^t \Rightarrow$$

$$2 = (1 + \frac{r}{100})^t$$

$$\ln 2 = t \cdot \ln (1 + \frac{r}{100}) \Rightarrow$$

$$t = \frac{\ln 2}{\ln (1 + \frac{r}{100})} \Rightarrow$$

$$t = \frac{0.693}{\frac{r}{100}} \Rightarrow$$

$$t = \frac{0.72}{\frac{r}{100}} \Rightarrow$$

$$t = \frac{72}{r}$$

The last few steps are done using the approximations $0.693 \approx 0.72$ and $\ln 1 + x \approx x$ for small values of x. We are able to do this as the rule of 72 is only an approximation.

3. $t_h = -\frac{\ln \frac{1}{2}}{k} = \frac{0.693}{k}$

4.

$$\frac{dy}{dt} = y(100 - y) \Rightarrow$$

$$\frac{1}{y(100 - y)} dy = dt$$

$$\frac{1}{100} \int \frac{1}{y} + \frac{1}{100 - y}\, dy = \int dt \Rightarrow$$

$$\frac{1}{100} \ln \frac{y}{100 - y} = t + c \Rightarrow$$

$$e^{\ln \frac{y}{100 - y}} = e^{100t + 100c} \Rightarrow$$

$$\frac{y}{100 - y} = Ce^{100t \Rightarrow}$$

$$y = 100Ce^{100t} - yCe^{100t} \Rightarrow$$

$$y + yCe^{100t} = 100Ce^{100t} \Rightarrow$$

$$y = \frac{100Ce^{100t}}{1 + Ce^{100t}}$$

Now that we have the general solution, we can plug in $y_0 = 5$ and $t = 5$ to find the value of C.

$$(5) = \frac{100Ce^{100(0)}}{1 + Ce^{100(0)}} \Rightarrow$$

$$5 = \frac{100C}{1 + C} \Rightarrow$$

$$5 + 5C = 100C \Rightarrow$$

$$5 = 95C \Rightarrow$$

$$C = \frac{1}{19}$$

The final solution is

$$y = \frac{100e^{100t}}{19(1 + (\frac{1}{19})e^{100t})}$$

7.15 JNB Lab: Calculus of Entropy in Daily Life and Monte Carlo Simulations

The goal of this lab is to explore how calculus can be used to explore the concept of entropy as might be useful within a variety of situations.

7.15.1 Derivative Optimization

Given a function $y = f(x)$, the derivative $f'(x)$ evaluated at $x = a$ gives us the slope of the tangent (instantaneous rate of change) to the graph of $y = f(x)$ and the point $(a, f(a))$. Maximum and minimum points are special since the tangents are horizontal at those points and hence the derivative satisfies the equation $f'(x) = 0$. Maximum and minimum points are important in applications since, for example, minimizing cost or maximizing the benefit of a project is often a primary goal.

Entropy is an important example discussed in the section on societal applications in the final chapter (complex systems) of this book. We illustrate here why minimization/maximization of entropy is important. Suppose you have a committee of N people voting on a certain decision such that a proportion $p_1 = p$ strongly prefers option A and a proportion $p_2 = 1-p$ strongly prefers option B . Note that there is no disagreement if $p_1 = 1$ and $p_2 = 0$ (everyone strongly prefers option A) and likewise there is no disagreement if $p_1 = 0$ and $p_2 = 1$ (everyone strongly prefers option B.) Note that if $p_1 = .9$ and $p_2 = .1$, there is a strong consensus for Option A so while there is a little bit of disagreement, Option A will likely carry. Note that the case where the population is evenly split $p_1 = p_2 = .5$ will have the greatest disagreement in trying to achieve a consensus.

Entropy can be used to measure the level of disagreement. We define the Shannon entropy as

$$H(p) = -\sum_{i=1}^{2} p_i \ln p_i = -[p \ln p + (1-p)\ln(1-p)].$$

where $0 \leq p \leq 1$. Exercise 1.1 shows the properties of this function $H(p)$ correspond to the amount of disagreement in our example.

Exercises

Exercises

1.1

a) Make a plot of $H(p) = -[p \ln p + (1-p)\ln(1-p)]$ on the interval $0 < p < 1$

b) Use L'Hospital's rule to evaluate the one-sided limits $H(0) = \lim_{p \to 0^+} H(p)$ and $H(1) = \lim_{p \to 1^-} H(p)$.

c) Use calculus to prove that a maximum value of $H(p)$ occurs at $p = .5$.

7.15.2 Series

Suppose now there are k options. p_i is the proportion of committee members favoring option i ($1 \leq i \leq k$ and $p_1 + ... + p_k = 1$).

$$H = -\sum_{i=1}^{k} p_i \ln p_i = -(p_1 \ln p_1 + p_2 \ln p_2 + ... + p_k \ln p_k).$$

Exercises

Exercises

2.1

1. Consider the case with $k = 3$. Compute the entropy H if

   a) $p_1 = .5, p_2 = .3, p_3 = .2$ and

   b) $p_1 = .2, p_2 = .5, p_3 = .3$

   c) $p_1 = .2, p_2 = .3, p_3 = .5$

2. What does problem 1 tell us about the computation of entropy of disagreement?

7.15.3 Probability Distributions

Now let us consider a situation where you are holding a dinner and there is a certain probability p_k that a guest prefers to arrive between k and $k + 1$ hours late. If all guests prefer to arrive no more than 1/2 hour late, then $p_0 = 1$ and all other $p_k = 0$, so the entropy $H = 0$. This is in certain cultures an ideal situation and there is no major disagreement in the preferred arrival time ($H = 0$).

In other cases, it might be normative for guests to arrive substantially later than the announced time. For example, if all guests prefer to arrive between $k = 1$ and $k = 2$ (between 1 and 2 hours late), then $p_1 = 1$ and again there is no major disagreement ($H = 0$).

We can use a probability distribution to model the probability of a preferred arrival time. For example, if the mean arrival time is 10 minutes after the designated time (1/k=1/6 hour), then we consider the probability density function $f(x) = \frac{1}{6}e^{-6x}$ where $x \in [0, \infty)$. The probability that a guest arrives in the interval $k \leq x \leq k + 1$ is given by the definite integral

$$p_k = prob(k \leq x \leq k + 1) = \int_k^{k+1} f(x)\, dx = (1/6) \int_k^{k+1} e^{-6x}\, dx = -\frac{1}{36} e^{-6x} \,|_k^{k+1} = -\frac{1}{36}[e^{-6(k+1)} - e^{-6k}].$$

For example, $p_0 = -\frac{1}{36}[e^{-6} - 1] \approx .9999$

and

$p_1 = -\frac{1}{36}[e^{-6(2)} - exp(-6)] \approx .0001$

so

$H \approx .001$. (There is almost no disagreement in preferred arrival time.)

7.15.4 Exercises

Exercises

3.1 An exponential distribution has the form $f(x) = ke^{-kx}$ where $k > 0$ and $0 \leq x < \infty$.

a) Show that $prob(0 \leq x < \infty) = 1$.

b) Given a continuous probability distribution $f(x)$ for $x \in I$, the mean is computed as $\int_{x \in I} xf(x)\,dx$. Find the mean for the exponential distribution in terms of k

7.15.5 Monte Carlo Simulation

Let us consider the context of teaching a class of math students with a random variable X denoting a standardized entrance test score. If $X < 0$, the score is below average, and if $X > 0$, the score is above average. Note that if all students are average, teaching the class is simpler than if there is a big spread in the students' range of ability. Entropy in this case is interpreted in this case to mean complexity.

Here we would like to compare two distributions with domain $-\infty < x < \infty$:

$f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2}$ (standard unit normal distribution)

$g(x) = \frac{1}{\pi} \frac{1}{1+x^2}$ (Cauchy distribution)

In particular, we would like to compare the contribution to the complexity (entropy) for the tails $x < -3$ and $x > 3$. We will classify students as 'usual' if $-3 \leq X \leq 3$ and `exceptional' if $x > 3$ or $X < -3$.

In practical terms, we are asking how the complexity is impacted by very strong or very weak students for these two distributions.

Intuitively we know for a normal distribution, there is only a .3% chance of being more than 3 standard deviations above or below the mean. So for a class, say of 10 students, we would expect .03 students to be exceptional. So there would be essentially no contribution to the complexity from a perspective focused on tangible results.

On the other hand, for a Cauchy distribution the probability is .2 to get a very strong or a very weak student, and so for a class of 10 students, we would expect to see 2 exceptional students. The contribution to complexity is $H_{tail} = -.2ln(.2) \approx .1$.

To estimate the complexity of a normally distributed vs. Cauchy distributed class, we can use a Monte Carlo simulation. That is, we draw a sample of 10 random values, determine the proportion of usual and exceptional students, and compute the entropy. We illustrate this for a routine class (normally distributed) and ask you to analyze a Cauchy distributed class.

```
Entropy of Normal Class= 0.008372164361174893
Contribution to Entropy by Exceptional Students= 0.005926806596313439
```

Exercises

Exercises

4.1

a) Sketch the unit normal and Cauchy distributions on the interval $-5 \leq x \leq 5$. What do you note about the tails of these distributions?

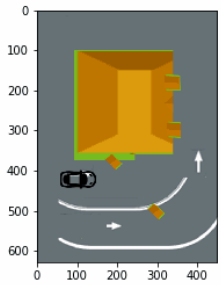
b) Do a Monte Carlo simulation with 10,000 trials to estimate the entropy of a Cauchy distributed class with 10 students as well as the contribution to the entropy by exceptional students.

Note

Please visit the web version of this section to engage with this interactive resource.

7.15.6 Random Number Simulation of a Drive-thru

The goal of this lab is to show how random numbers are used to simulate a scenario where the owner of a fast-food restaurant is contemplating whether or not to open a second drive-thru lane as illustrated in the animation below. (For an explanation of how to create this animation, see the section on object-oriented programming in the chapter Overview of Python Programming.) The section on Chicago homicides gives a related example of how random numbers can be used in "Monte Carlo simulation".

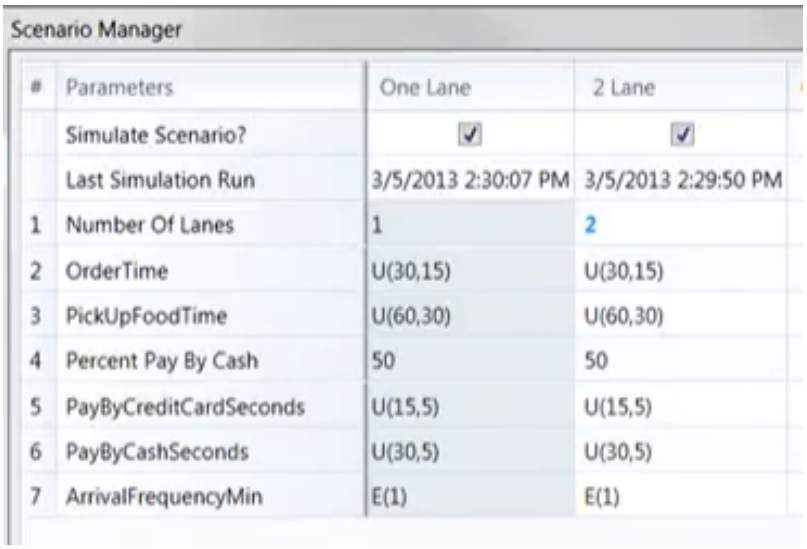


Our main reference for this modeling problem is described in this video

"Fast Food Drive Thru 2 lanes versus 1" YouTube, uploaded by Daniel Hickman, 5 March 2013, https://www.youtube.com/watch?v=M9v6bx3l4Dk. Permissions: YouTube Terms of Service

7.15.7 Model Distributions

A drive-thru model begins with empirical data which can be described using *probability density functions (PDFs)*, also called *distributions*. We use the same distributions given in Hickman's video as summarized below:



Scenario Manager

#	Parameters	One Lane	2 Lane
	Simulate Scenario?	☑	☑
	Last Simulation Run	3/5/2013 2:30:07 PM	3/5/2013 2:29:50 PM
1	Number Of Lanes	1	2
2	OrderTime	U(30,15)	U(30,15)
3	PickUpFoodTime	U(60,30)	U(60,30)
4	Percent Pay By Cash	50	50
5	PayByCreditCardSeconds	U(15,5)	U(15,5)
6	PayByCashSeconds	U(30,5)	U(30,5)
7	ArrivalFrequencyMin	E(1)	E(1)

We will explain a value r generated by a PRNG (pseudo-random number generator) can be converted into

- A=length of time before the next car arrives at the parking lot
- O=length of time to place an order
- P=length of time to pay for the food
- F=length of time to pick up food

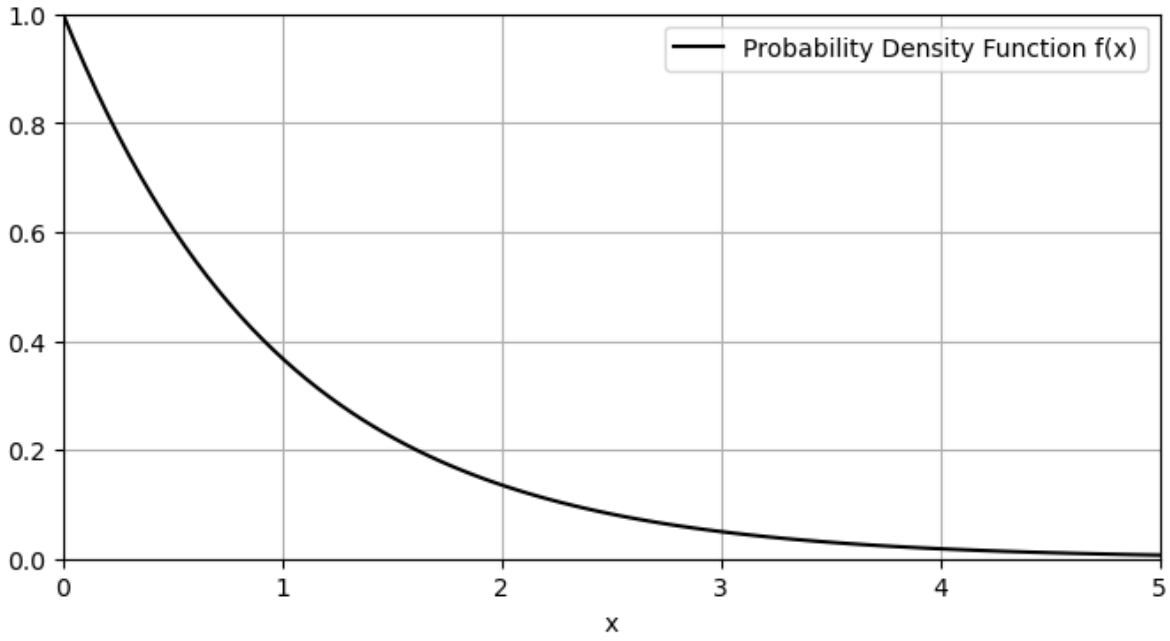
A=Length of time before next arrival

Let X be a random variable giving the length of time (in minutes) until the next car arrives at the parking lot. We assume X is an exponential distribution with mean $1/k = 1$. In this case, the probability density function (pdf) is

$$f(x) = ke^{-kx} = e^{-x}$$

Here is a graph of this pdf:

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0,5,100)
f= np.exp(-x)  #standard (unit) normal distribution
plt.figure(figsize=(8, 4))
plt.plot(x,f,color='k',label="Probability Density Function f(x)")
plt.xlim((0,5))
plt.ylim((0,1))
plt.xlabel("x")
plt.grid()
plt.legend()
plt.show()
```

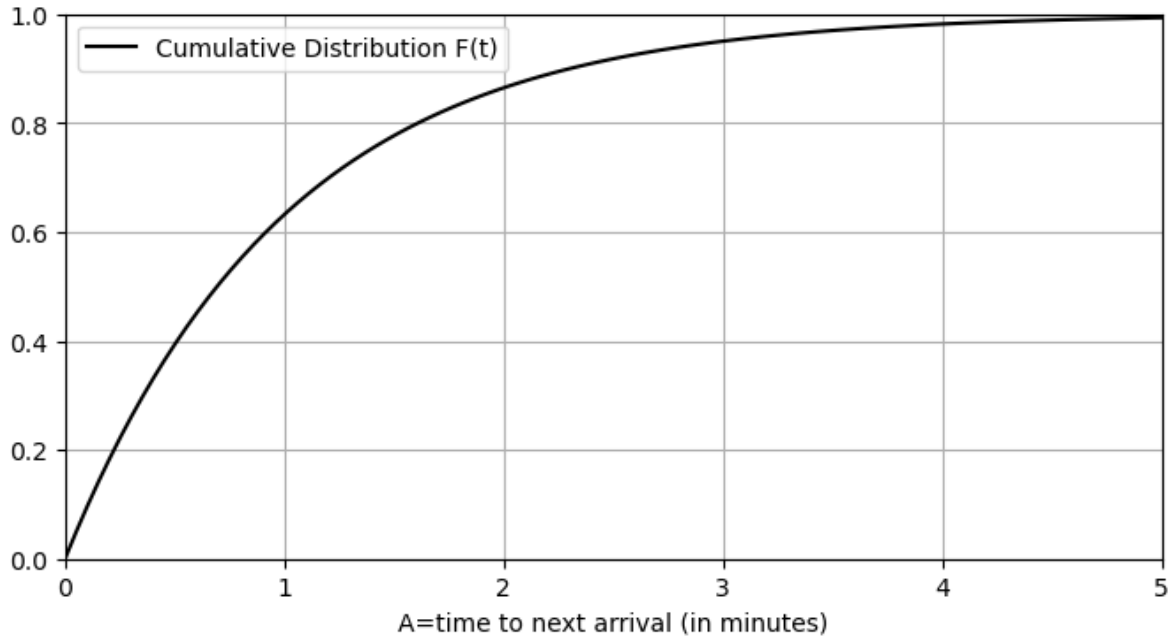


The corresponding cumulative distribution function (cdf) is

$$F(t) = prob(0 \leq X \leq t) = \int_0^t ke^{-kx}\,dx = -e^{-kx}\,|_0^t = -e^{-kt} + 1 = -e^{-t} + 1.$$

Here is a graph of this cdf:

```
import numpy as np
import matplotlib.pyplot as plt
t = np.linspace(0,5,100)
F= 1-np.exp(-t)  #standard (unit) normal distribution
plt.figure(figsize=(8, 4))
plt.plot(t,F,color='k',label="Cumulative Distribution F(t)")
plt.xlim((0,5))
plt.ylim((0,1))
plt.xlabel("A=time to next arrival (in minutes)")
plt.grid()
plt.legend()
plt.show()
```



Note that the $F(t) = 1 - e^{-t}$ is an increasing function with a value of 0 at $t = 0$ and a range $[0, 1)$. As such, each random number r $(0 \leq r < 1)$ corresponds to a unique value of A:

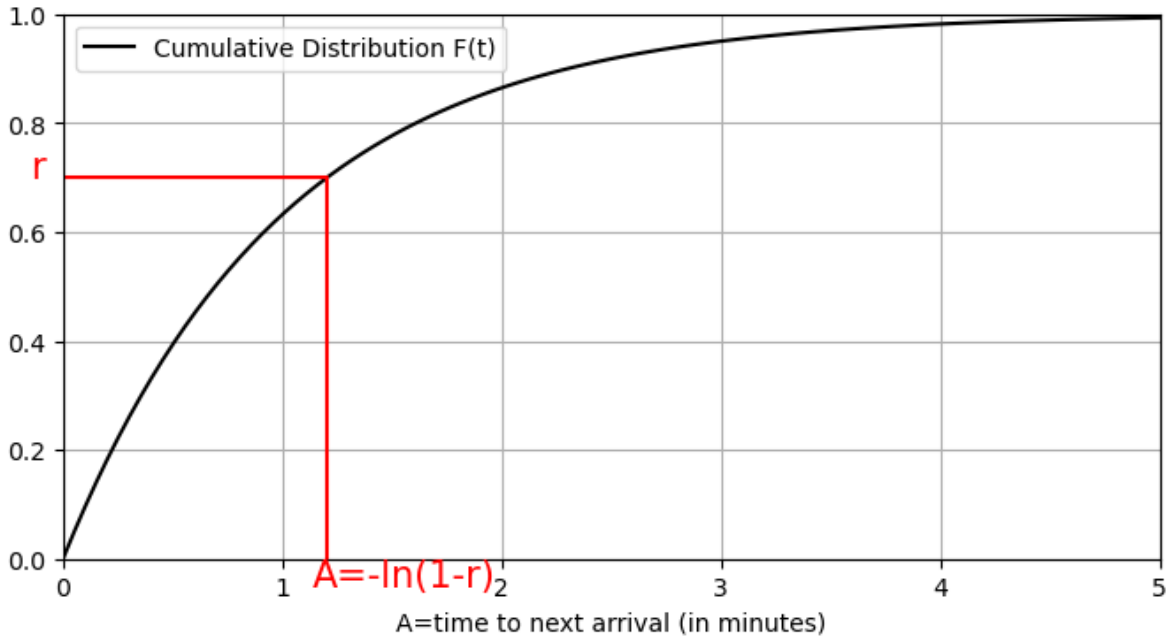
$$r = 1 - e^{-kA} \Rightarrow$$

$$e^{-kA} = 1 - r \Rightarrow$$

$$-kA = \ln(1 - r) \Rightarrow$$

$$A = -\frac{1}{k}\ln(1 - r) = -\ln(1 - r).$$

```
import numpy as np
import matplotlib.pyplot as plt
t = np.linspace(0,5,100)
F= 1-np.exp(-t)  #standard (unit) normal distribution
plt.figure(figsize=(8, 4))
plt.plot(t,F,color='k',label="Cumulative Distribution F(t)")
plt.xlim((0,5))
plt.ylim((0,1))
plt.text(-.15,.7,"r",color='r',size=15)
plt.text(-np.log(1-.7)-.07,-.05,"A=-ln(1-r)",color='r',size=15)
plt.plot([0,-np.log(.3),-np.log(.3)],[.7,.7,0],color='r')
plt.xlabel("A=time to next arrival (in minutes)")
plt.grid()
plt.legend()
plt.show()
```

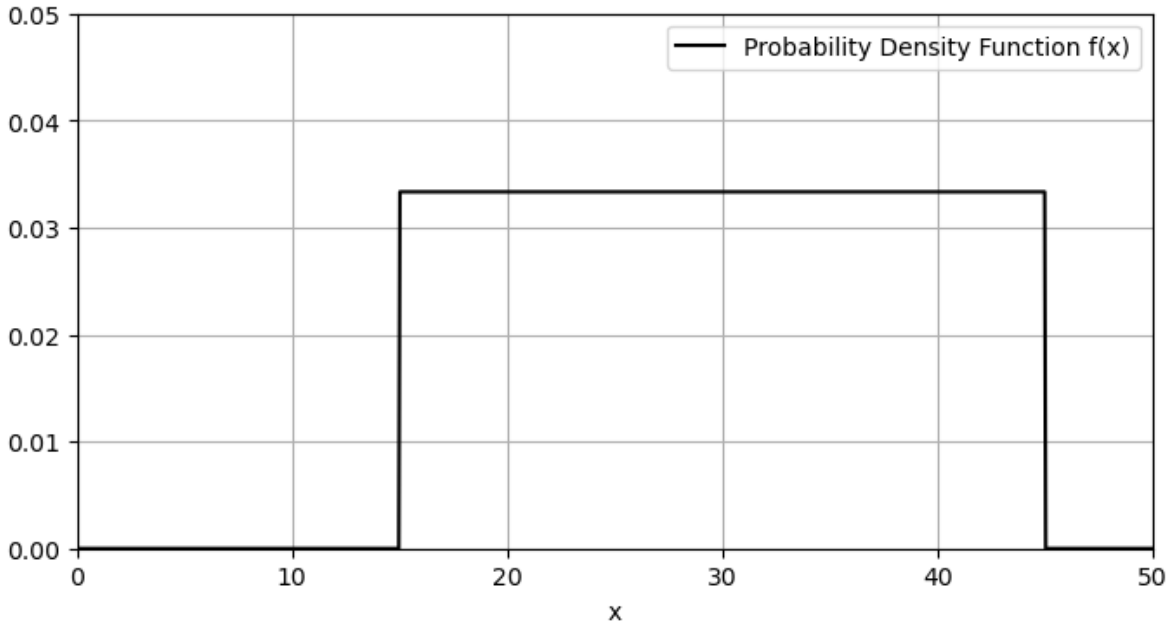


O=length of time to place an order

Let X be a random variable giving the length of time (in seconds) to place an order. We assume X is uniformly distributed on the interval [15,45] (denoted U(30,15) where 30 is the length of the interval and 15 is the minimum value for X.) In this case, the probability density function (pdf) is

$$f(x) = \frac{1}{30} \ (15 \leq x \leq 45)$$

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0,50,1000)
f= (1/30)*(np.heaviside(x-15,0)-np.heaviside(x-45,0))    #standard (unit) normal␣
↪distribution
plt.figure(figsize=(8, 4))
plt.plot(x,f,color='k',label="Probability Density Function f(x)")
plt.xlim((0,50))
plt.ylim((0,.05))
plt.xlabel("x")
plt.grid()
plt.legend()
plt.show()
```

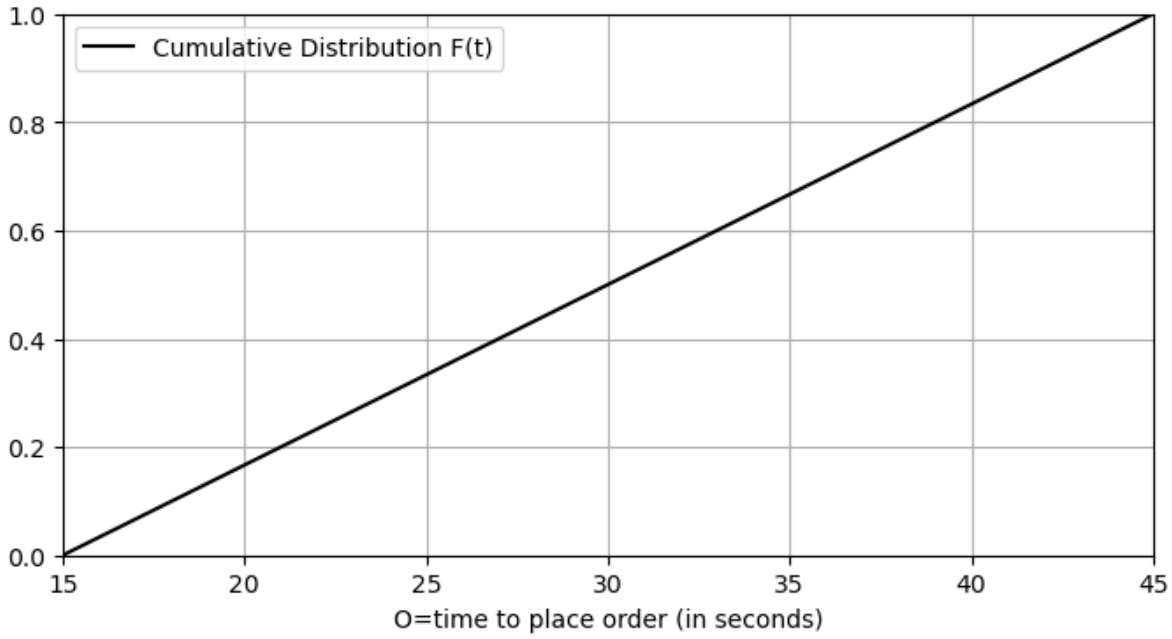


The cumulative distribution function (cdf) is

$$F(t) = prob(15 \leq X \leq t) = \int_0^t \frac{1}{30}\, dx = -\frac{1}{30} x \,|_{15}^{t} = \frac{t}{30} - \frac{15}{30} = \frac{t-15}{30}$$

Here is a graph of this cdf:

```
import numpy as np
import matplotlib.pyplot as plt
t = np.linspace(15,45,100)
F= ((t-15)/30)*np.heaviside(t-15,0)  #U(30,15)
plt.figure(figsize=(8, 4))
plt.plot(t,F,color='k',label="Cumulative Distribution F(t)")
plt.xlim((15,45))
plt.ylim((0,1))
plt.xlabel("O=time to place order (in seconds)")
plt.grid()
plt.legend()
plt.show()
```

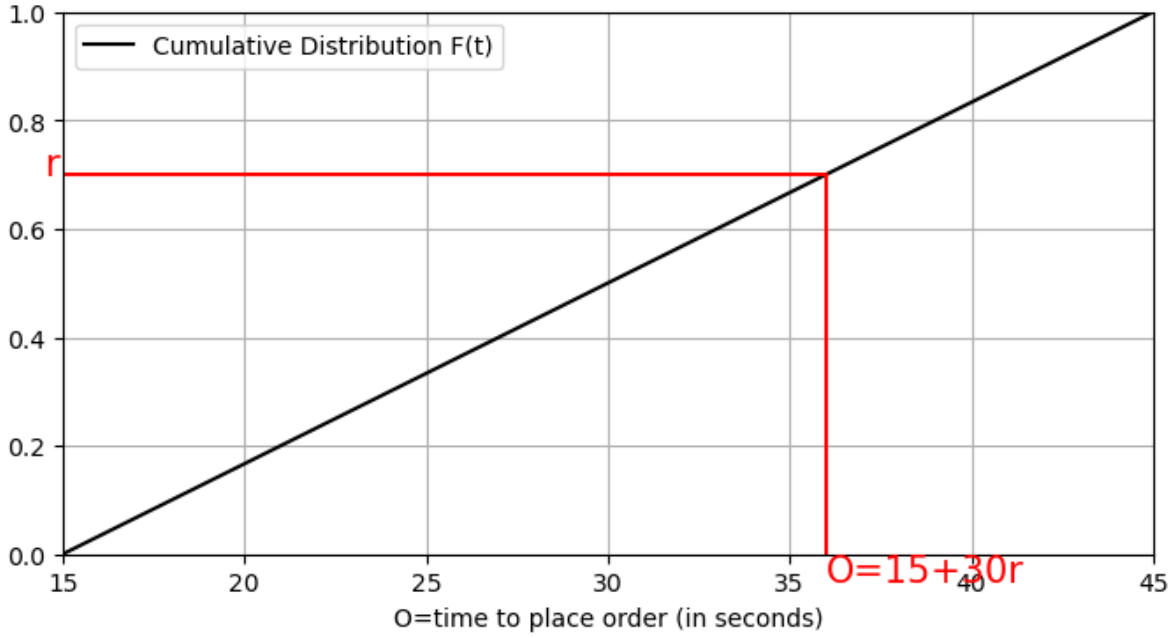


As before, the $F(t) = \frac{t-15}{30}$ $(15 \leq t \leq 45)$ is an increasing function with a value of 0 at $t = 15$ and a range $[0, 1)$. As such, each random number r $(0 \leq r < 1)$ corresponds to a unique value of O where $15 \leq O \leq 45$:

$$r = \frac{O - 15}{30} \Rightarrow$$

$$O - 15 = 30r \Rightarrow$$

$$O = 15 + 30r.$$



Similarly, we can convert a random number r into a length of time P to pay for food or a length of time F to pick up food.

P=length of time to pay for the food

By Cash

Let X be a random variable giving the length of time (in seconds) to pay for an order using cash. We assume X is uniformly distributed on the interval [5,35] (denoted U(30,5) where 30 is the length of the interval and 5 is the minimum value for X in seconds.) In this case, the probability density function (pdf) is

$$f(x) = \frac{1}{30} \ (5 \leq x \leq 35)$$

The cdf is

$$F(t) = \frac{1}{30}(t-5)$$

$(5 \leq t \leq 35)$.

Given a random number r $(0 \leq r \leq 1)$, the pay by cash time is $P = 30r + 5$.

By Credit

Let X be a random variable giving the length of time (in seconds) to pay for an order using a credit card. We assume X is uniformly distributed on the interval [5,20] (denoted U(15,5) where 15 is the length of the interval and 5 is the minimum value for X in seconds.) In this case, the probability density function (pdf) is

$$f(x) = \frac{1}{15} \ (5 \leq x \leq 20)$$

The cdf is

$$F(t) = \frac{1}{15}(t-5)$$

$(5 \leq t \leq 20)$.

Given a random number r $(0 \leq r \leq 1)$, the pay by cash time is $P = 15r + 5$.

F=length of time to pick up food

Let X be a random variable giving the length of time (in seconds) to pick up an order. We assume X is uniformly distributed on the interval [30,90] (denoted U(60,30) where 60 is the length of the interval and 30 is the minimum value for X in seconds.) In this case, the probability density function (pdf) is

$$f(x) = \frac{1}{60} \ (30 \leq x \leq 90)$$

The cdf is

$$F(t) = \frac{1}{60}(t-30)$$

$$(30 \leq t \leq 90).$$

Given a random number r $(0 \leq r \leq 1)$, the pay by cash time is $P = 60r + 30$.

Exercises

5.1

a) Use random numbers to simulate arrivals at a one-lane drive-thru. For simplicity, assume that:

the time A between arrivals is exponentially distributed with mean k=1 (in minutes);

the order time O is uniformly distributed on the interval $[15, 45]$ (in seconds);

the pay time P is uniformly distributed on the interval $[5, 25]$ (in seconds); and

the food pick-up time F is uniformly distributed on the interval $[30, 60]$ in seconds.

Find the mean total time to enter and be processed through the order-pay-pickup system.

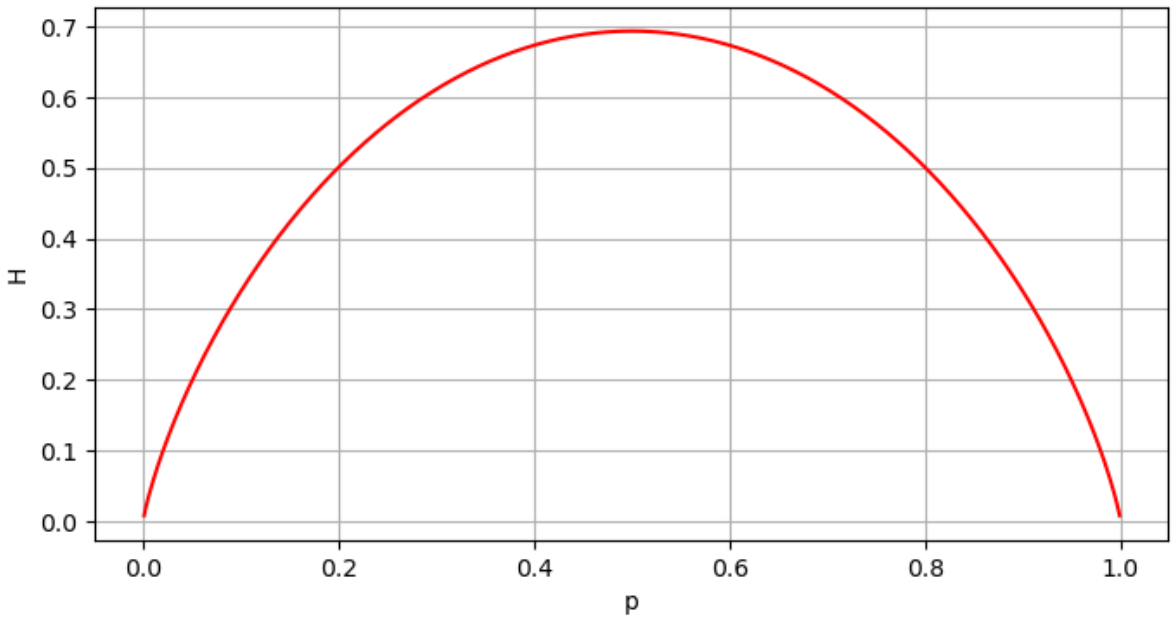
Be sure to convert the time between arrivals to seconds and give the final mean total time in minutes.

b) Explore how the mean total time depends on system parameters. For example, you might vary the parameter k in the arrival time from k=.1 (6 seconds) to 2 (2 minutes) in increments of .1

7.16 JNB Lab Solutions

Exercise 1.1

a)



b) Note that $H(p) = H(1-p)$, so $\lim_{p\to 0^+} H(p) = \lim_{p\to 1^-} H(p)$.

Using L'Hospital's Rule,

$$\lim_{-p\to 0^+} p\ln p = \lim_{p\to 0^+} \frac{-\ln p}{1/p} = \lim_{p\to 0^+} \frac{-1/p}{-1/p^2} = \lim_{p\to 0^+} p = 0.$$

Furthermore, $\lim_{p\to 0^+} -(1-p)\ln(1-p) = \lim_{q\to 1^-} -q\ln q = 0$. The desired result follows.

c)

Let $H(p) = -[p \ln p + (1-p)\ln(1-p)]$. Then

$$H'(p) = -[(\ln p + 1) + (-\ln(1-p) - 1) = -[\ln p - \ln(1-p)] = -[\ln \frac{p}{1-p}] = 0 \Rightarrow \frac{p}{1-p} = 1 \Rightarrow p = 1 - p \Rightarrow p = .5.$$

Moreover, for values of p near $p = .5$, if $p < .5$ then $H'(p) > 0$ so H is increasing, and if $p > .5$, then $H'(p) < 0$ so p is decreasing. This implies $H(.5)$ is a max.

Note that in this case $H''(.5) = 0$ so the second derivative test does not apply.

Exercise 2.1

1. a), b), and c) all have the same value of H= -(.5 ln .5 + .3ln .3 + .2 ln .2) $\approx$ 1.03.
2. Which option is labeled "1", "2" or "3" is inconsequential in the computation of the entropy of disagreement. (Note that this makes it difficult to apply entropy to a histogram where the bin numbers are purposefully ordered. For example, the label i might mean a student fails i tests in a particular math course.)

Exercise 3.1

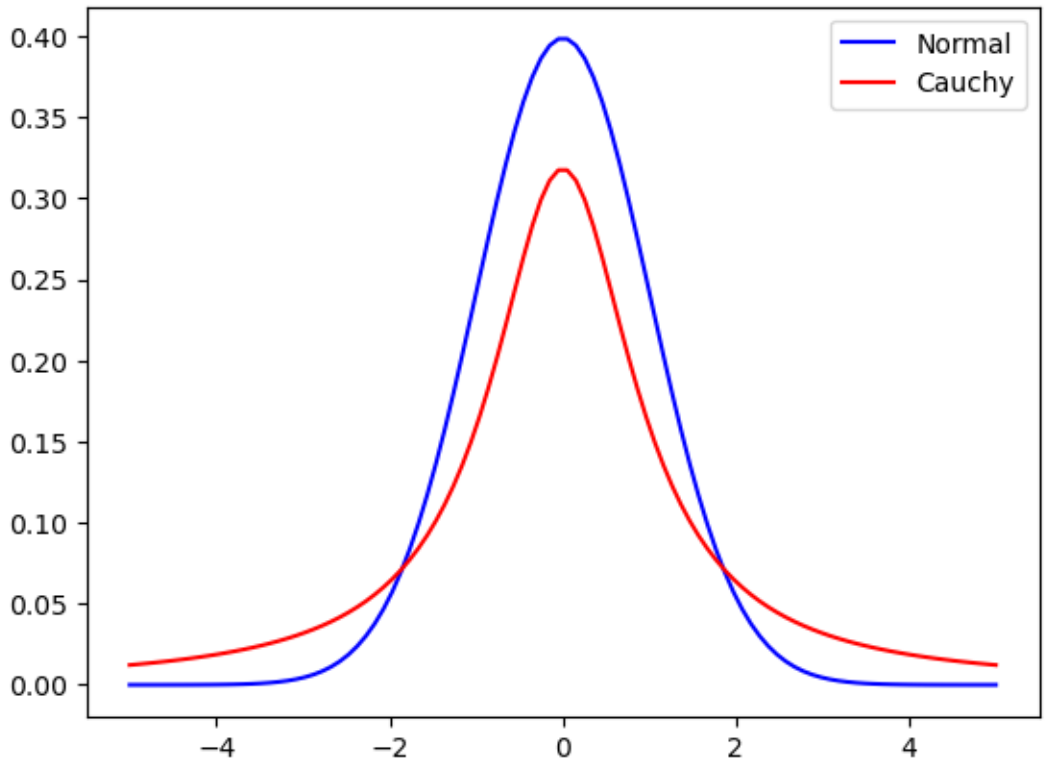
a) $\int_0^\infty ke^{-kx}\,dx = -e^{-kx}\,|_0^\infty = 0 + 1 = 1.$

b) The mean $E(X)$ is given by the integral $E(X) = \int_0^\infty (kx)e^{-kx}\,dx$. Using integration by parts with $u = kx$, $du = k\,dx$, $dv = e^{-kx}\,dx$, $v = -(1/k)e^{-kx}$, we have

$E(X) = -xe^{-kx}\,|_{x=0}^\infty + \int_{x=0}^\infty -e^{-kx}\,dx = 0 - (1/k)e^{-kx}\,|_{x=0}^\infty = 1/k.$

Exercise 4.1

a)



b)

```
Entropy of Cauchy Class= 0.4498874005479002
Contribution to Entropy by Exceptional Students= 0.27849303712298606
```

Exercise 5.1

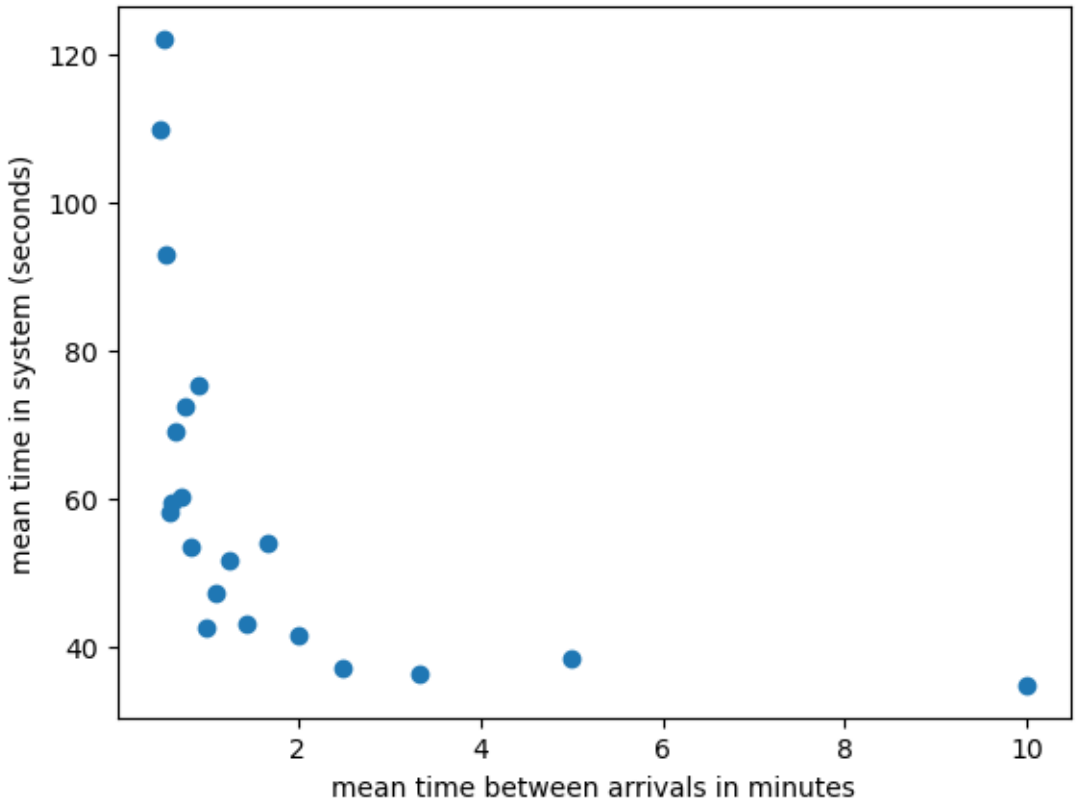
```
table = df[["order_arrival_time","time_to_order","order_system_time","pay_arrival_time
 ↪","time_to_pay","pay_system_time","pickup_arrival_time","time_to_pickup","pickup_
 ↪system_time", "finish_time"]]
table
```

	order_arrival_time	time_to_order	order_system_time	pay_arrival_time	\
0	0.000000	43.602984	43.602984	43.602984	
1	481.388415	33.366122	33.366122	514.754537	
2	1219.912249	41.477045	41.477045	1261.389293	
3	1294.706715	30.565060	30.565060	1325.271775	
4	1683.852787	43.093286	43.093286	1726.946073	
5	2240.226570	43.522066	43.522066	2283.748635	
6	2859.137294	39.779653	39.779653	2898.916947	
7	4292.462366	27.259678	27.259678	4319.722044	
8	4423.265534	35.461872	35.461872	4458.727406	
9	5183.852133	22.406112	22.406112	5206.258245	

	time_to_pay	pay_system_time	pickup_arrival_time	time_to_pickup	\
0	7.556623	7.556623	51.159607	65.477791	
1	6.231221	6.231221	520.985758	56.592074	
2	23.202241	23.202241	1284.591534	58.655755	
3	12.672773	12.672773	1337.944548	43.237563	
4	6.338934	6.338934	1733.285006	82.937953	
5	16.409112	16.409112	2300.157747	51.509599	
6	6.035139	6.035139	2904.952086	57.808210	
7	23.996465	23.996465	4343.718509	76.043934	
8	34.086612	34.086612	4492.814018	76.186779	
9	8.030864	8.030864	5214.289109	37.802962	

	pickup_system_time	finish_time
0	65.477791	116.637398
1	56.592074	577.577832
2	58.655755	1343.247289
3	48.540305	1386.484852
4	82.937953	1816.222959
5	51.509599	2351.667346
6	57.808210	2962.760296
7	76.043934	4419.762443
8	76.186779	4569.000797
9	37.802962	5252.092071

```
Text(0, 0.5, 'mean time in system (seconds)')
```



Note: This type of analysis can be done using an Excel spreadsheet. For example, see https://docs.google.com/spreadsheets/d/1JPq2UfIWMTHizAq6_gH9WYm_qXZodi7v/edit?usp=sharing&ouid=101192945680365451790

CHAPTER 8

CALCULUS - Inne Singgih

> **Note**
>
> This chapter on Calculus was written by Dr. Inne Singgih (Department of Mathematical Sciences, University of Cincinnati). The lab on calculus animations was contributed by Dr. Paul Isihara.

8.1 Introduction

This Calculus Module offers a brief introduction to several important topics in undergraduate Calculus:

- Functions (Section 2)
- Rates, Limits, and Continuity (Section 3)
- Derivatives (Section 4)
- Integrals (Section 5)
- Parametric Equations (Section 6)
- Sequences and Series (Section 7)

For each topic, we provide basic definitions, examples, and exercises that are designed to be easily understood by entry-level undergraduate students. The exercises include problems students can solve manually to check their understanding of the concepts but are also solvable using Python Jupyter Notebooks.

This module focuses on the pure mathematical sides of Calculus, while the application can be found under the **Calculus for Daily Life** Module in the **APPLICATIONS** chapter of this book. In many cases, what prevents students from being successful in Calculus courses is that they are not yet proficient with the underlying algebra skills, which are not included in this Calculus Module. Students who are struggling in those areas can visit the **PRE-COLLEGE OUTREACH** chapter of this book.

8.2 Functions

Note

Below is the list of topics that are covered in this section:

- Function Definition and Terminology
- Function Behaviors
- Common Functions (Linear, Exponential, Logarithm, Logistic)
- Function Operations, Compositions, and Inverse

```
import sympy as sp
import numpy as np
import matplotlib.pyplot as plt
```

8.2.1 Function Definition and Terminology

A **function** is a *rule* that assigns exactly one output to each input. When a function f has input x, the output is written $f(x)$. This notation is read "f of x".

Example 1

- $f(x) = x + 3$ is a rule that adding 3 into the input x, giving the output $x + 3$.
- $g(x) = 8 - x^2$ is a rule that squaring the input x, then subtracting the result from 8, giving the output $8 - x^2$.

Code example for $f(x) = x + 3$ is given below. Enter an input value and the code will return the output value.

```
Enter input value x =1
The output is f(x) = 4.0
```

The **domain** of a function is the set of all input values that can be plugged into a function such that the output exists.

The **range** of a function is the set of all possible output values that a function can take.

Note that for the domain we need to avoid division by zero, square roots of negative numbers, and logarithms of nonpositive numbers. Can you explain why?

Python itself does not have built-in functionality to directly determine the domain and range of a given function. The domain and range of a function often depend on the specific characteristics of the function and its mathematical properties.

Example 2

Given $f(x) = \frac{1}{\sqrt{x}}$.

Excluding the zeroes of the denominator and negative number inside the square root, the domain of $f(x)$ is $(0, \infty)$. The range of $f(x)$ hence also $(0, \infty)$.

Code example for $f(x)$ is given below.

```
# Define the function symbolically
x = sp.Symbol('x')
f = 1 / (sp.sqrt(x))

# Find the domain: denominator not zero AND inside square root must be positive
domain = sp.solve(sp.denom(f) != 0 and x>0, x)

# Print the domain
print("Domain:", domain)
```

```
Domain: (0 < x) & (x < oo)
```

8.2.2 Function Behaviors

Increasing, Decreasing, and Constant Function

A function f defined over an interval is:

- **increasing** if the output values increase as the input values increase.
- **decreasing** if the output values decrease as the input values increase.
- **constant** if the output values remain the same as the input values increase.

Concave Up and Concave Down Function, Inflection Point

A function f defined over an interval is:

- **concave up** if a graph of f appears to be a portion of an arc opening upward.
- **concave down** if a graph of f appears to be a portion of an arc opening downward.

A point on a function changes concavity is called an **inflection point**.

8.2.3 Common Functions (Linear, Exponential, Logarithm, Logistic)

Linear Functions

A linear function is a function that can be expressed in the form $f(x) = mx + b$, where m represents the slope and b represents the y-intercept.

The slope m:

- Determines the steepness or inclination of the line. A positive slope indicates an upward trend, a negative slope indicates a downward trend, and a slope of zero represents a horizontal line.
- Represents the rate at which the function's output changes with respect to the input.
- Is calculated as the ratio of the change in y-coordinates to the change in x-coordinates between two points on the line, that is, between point (x_1, y_1) and (x_2, y_2), the slope $m = \dfrac{y_2 - y_1}{x_2 - x_1}$.

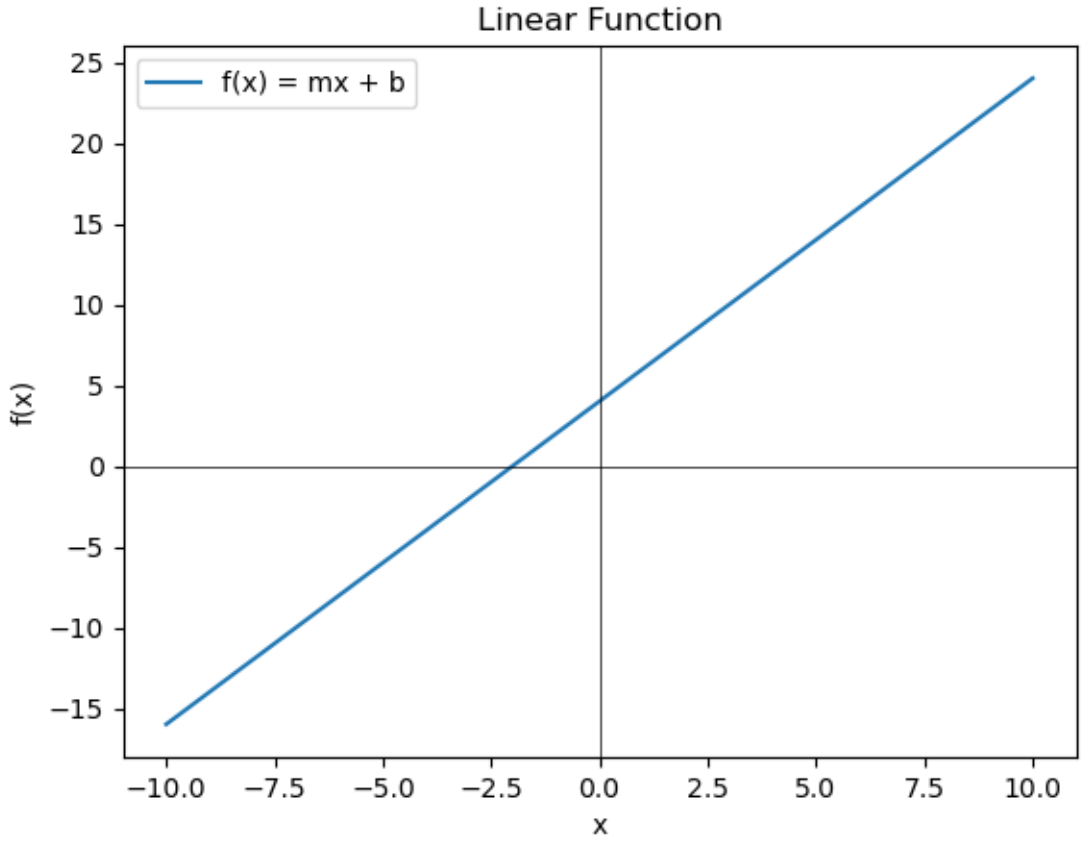
The y-intercept b:

- Is the point where the line intersects the y-axis.
- Represents the value of the $f(x)$ when $x = 0$.

To find the x-intercept, set $y = 0$ in the equation $f(x) = mx + b$ and solve for x.

Below is the code prompting user to input the constants m, and b.

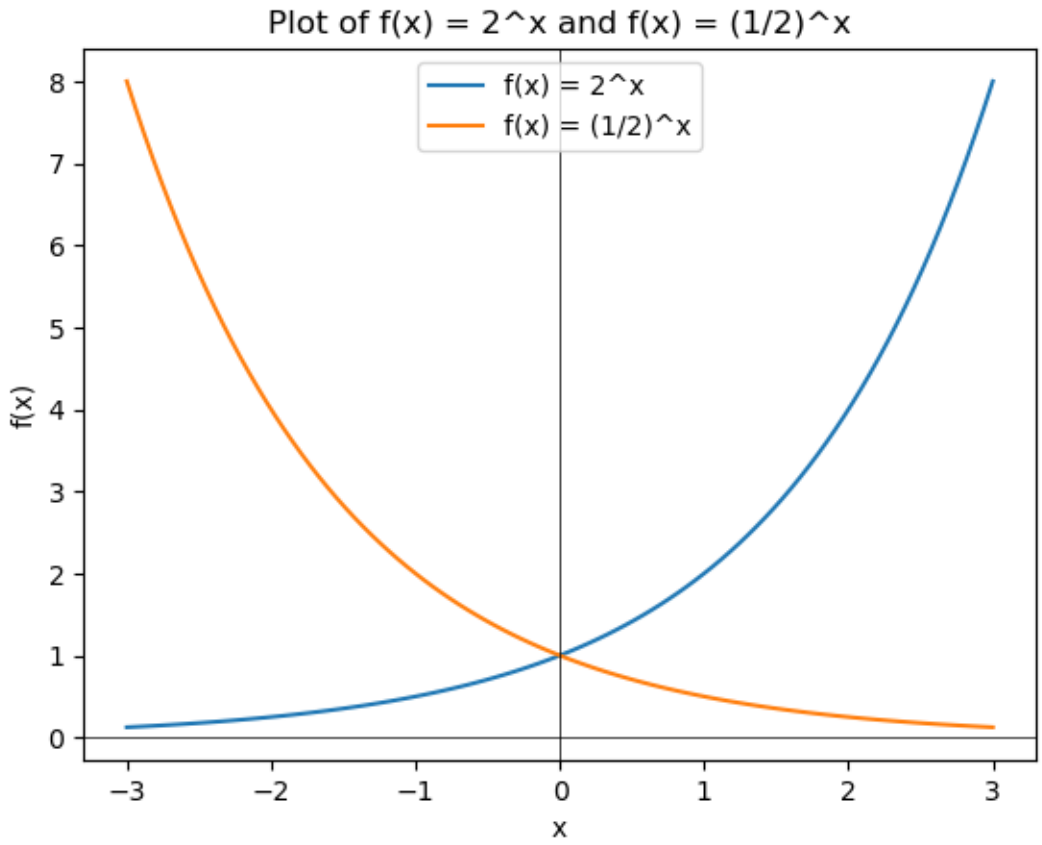
```
Enter the value for the slope (m): 2
Enter the value for the y-intercept (b): 4
```



Exponential Functions

An exponential function is a function in the form $f(x) = ab^x$, where a is the y-intercept and $b > 0$ is the constant multiplier.

Below is the code to plot the graphs of $f(x) = 2^x$ and $f(x) = \left(\frac{1}{2}\right)^x$:



Properties of $f(x) = ab^x$:

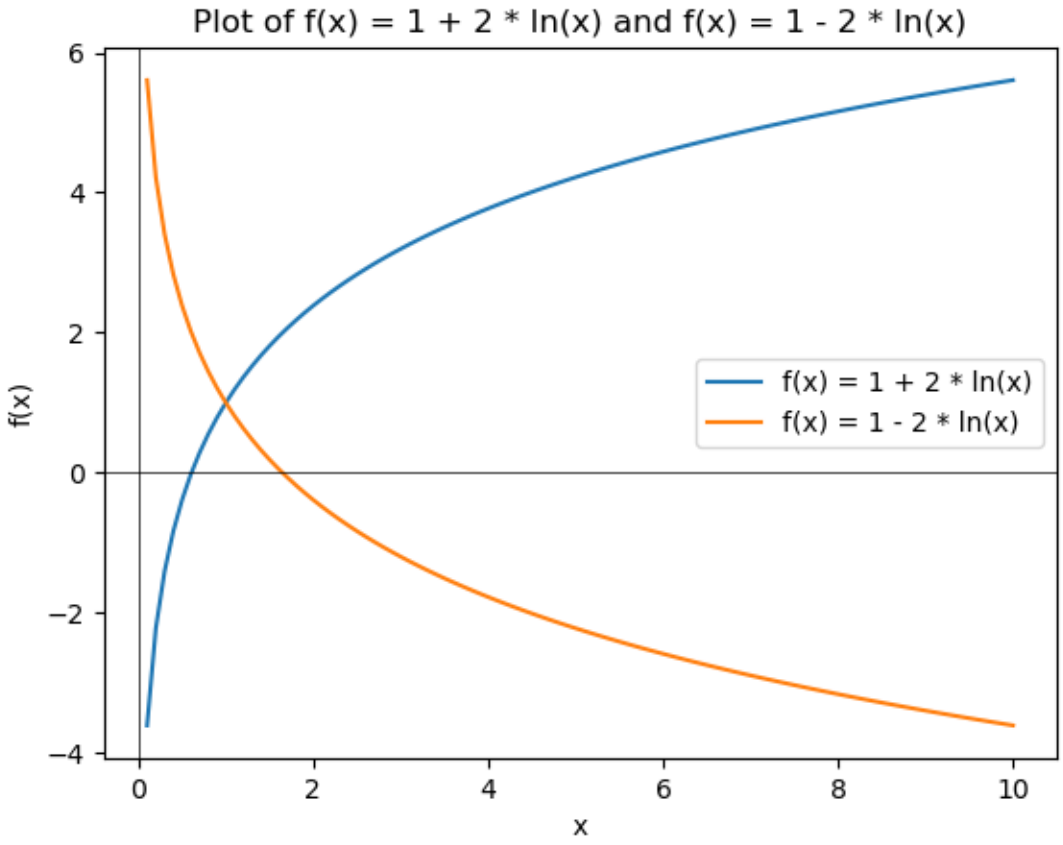
- $f(0) = a$. The y-intercept is always at $y = a$.
- $f(x) > 0$. An exponential function is always positive (never touch zero either)
- The domain of $f(x)$ is $(-\infty, \infty)$ while the range of $f(x)$ is $(0, \infty)$.
- If $0 < b < 1$ then
  - $f(x)$ goes to 0 as x goes to ∞.
  - $f(x)$ goes to ∞ as x goes to $-\infty$.
- If $b > 1$ then
  - $f(x)$ goes to ∞ as x goes to ∞.
  - $f(x)$ goes to 0 as x goes to $-\infty$.

The **natural exponential function** is $f(x) = e^x$, where e is the Euler's number $e = 2.71828182845905\ldots$

Logarithm Functions

A logarithm (log) function is a function in the form $f(x) = a + b\ln(x)$, where a and $b \neq 0$ are constants for positive input x.

Below is the code to plot the graphs of $f(x) = 1 + 2\ln(x)$ and $f(x) = 1 - 2\ln(x)$:



Properties of $f(x) = a + b\ln(x)$:

- The input x is restricted to be positive since $\ln(x)$ is only defined for $x > 0$.
- The log function then can be either increasing or decreasing depending on the value of b.
- For increasing log functions:
  - concave down
  - $b > 0$
  - vertical asymptote $x = 0$, with $f(x)$ goes to $-\infty$
  - $f(x)$ goes to ∞ as x goes to ∞
- For decreasing log functions:
  - concave up
  - $b < 0$
  - vertical asymptote $x = 0$, with $f(x)$ goes to ∞
  - $f(x)$ goes to $-\infty$ as x goes to ∞

Logistic Functions

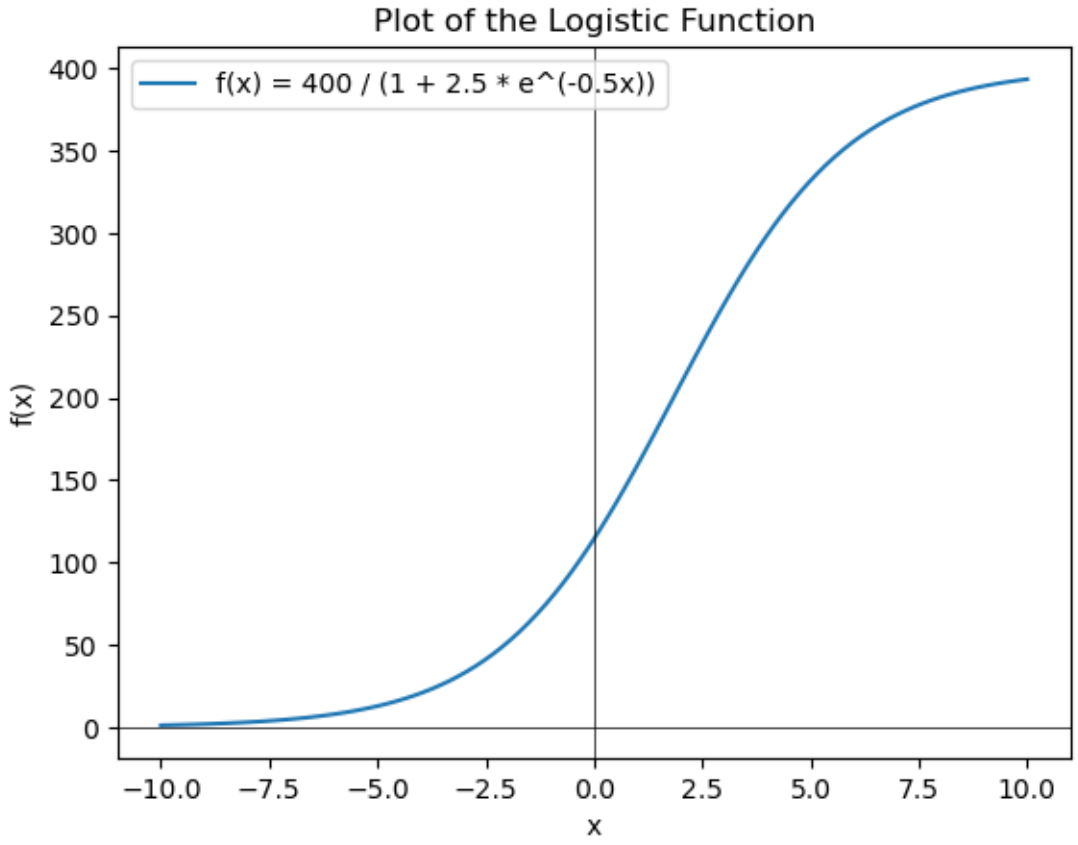
While exponential and log functions are commonly used in real life, it is unrealistic to believe that exponential growth can continue forever. In many situations, the growth ultimately restricted by the logistic availability.

A logistic function is a function in the form $f(x) = \frac{L}{1 + ae^{-bx}}$, where L, a, and b are nonzero constant.

In this function, $y = 0$ and $y = L$ are the horizontal asymptotes of the graph of $f(x)$, corresponding to the fact that the amount of resources is always positive but the growth is limited at a certain point.

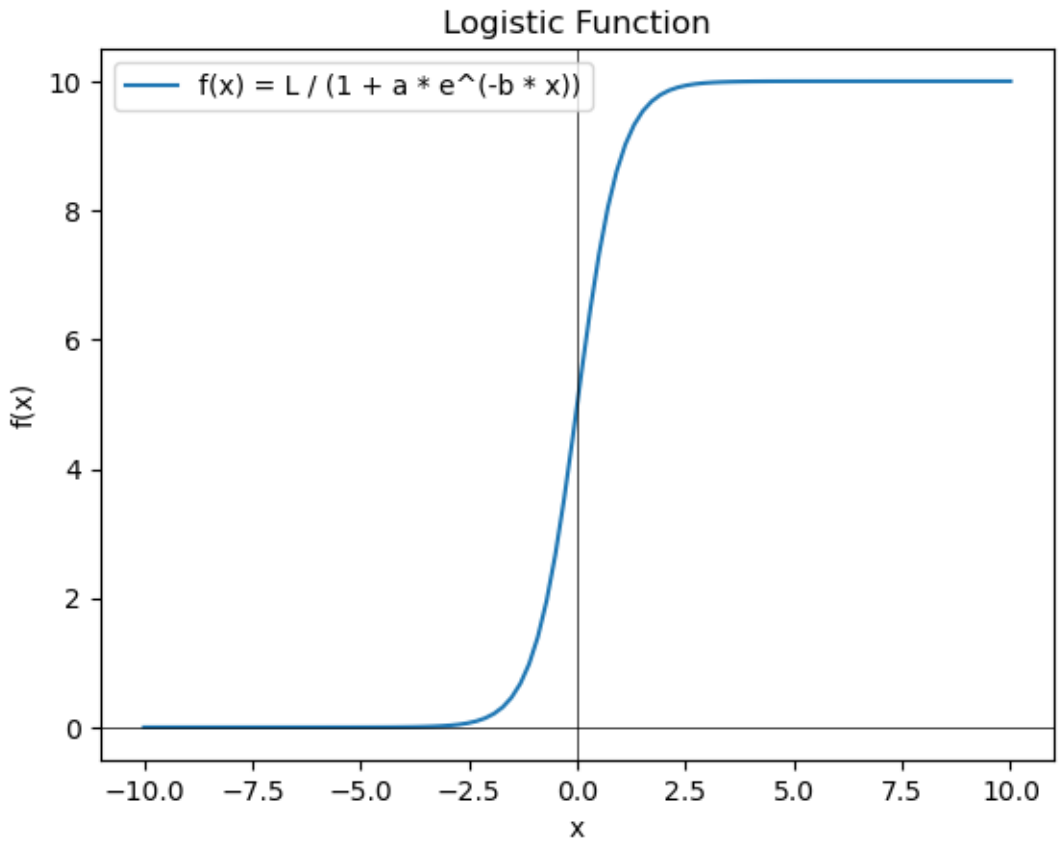
Depending on the positive or negative growth of the logistics, the logistic function can either increase or decrease.

Below is an example of the graph of an increasing logistic function $f(x) = \frac{400}{1 + 2.5e^{-0.5x}}$:



Below is the code prompting user to input the constants L, a, and b.

```
Enter the value for L: 10
Enter the value for a: 1
Enter the value for b: 2
```



Properties of $f(x) = \dfrac{L}{1 + ae^{-bx}}$:

- For increasing logistic function:
  - The function begins concave up then changes to concave down
  - $b > 0$
  - Horizontal asymptotes $y = 0$ to the left and $y = L$ to the right
- For decreasing logistic function:
  - The function begins concave down then changes to concave up
  - $b < 0$
  - Horizontal asymptotes $y = L$ to the left and $y = 0$ to the right

Both increasing and decreasing logistic functions have exactly one inflection point.

Without further contextual restriction on the domain, the domain of a logistic function is $(-\infty, \infty)$ and the range is $(0, L)$.

8.2.4 Function Operations, Compositions, and Inverse

Function Operations

Note that not everything can be mathematically operated. To show a simple example, adding apples and oranges does not equal an exclusive number of either apples or oranges. If given context, it is important to pay attention to what you are trying to operate. As mentioned in the Introduction section, we focus on the values rather than the contexts in this Calculus module.

If the input x of functions f and g are well-defined for the functions to be operated, then the following new functions can be constructed:

- **function addition:** $[f+g](x) = f(x) + g(x)$
- **function subtraction:** $[f-g](x) = f(x) - g(x)$
- **function multiplication:** $[f \cdot g](x) = f(x) \cdot g(x)$
- **function division:** $\left[\frac{f}{g}\right](x) = \frac{f(x)}{g(x)}$, where $g(x) \neq 0$

Example 3

Below is a code example that prompts the user to input a value of x, as well as two functions $f(x)$ and $g(x)$.

It then computes and returns the sum, difference, product, and quotient of the functions at the given value of x.

Note that $g(x)$ cannot be zero at the given input x.

```
Enter the value of x: 3
Enter the first function (f(x)): x**2
Enter the second function (g(x)): 3*x-1
Sum: 17.0
Difference: 1.0
Product: 72.0
Quotient: 1.125
```

Function Compositions

The **function composition** of two functions is evaluated by plugging one function into the other function. The composition operator is $\circ$.

- $(f \circ g)(x) = f(g(x))$
- $(g \circ f)(x) = g(f(x))$

Example 4

Given $f(x) = 3x^2 + 10$ and $g(x) = 1 - 2x$.

- $(f \circ g)(x) = f(g(x)) = f(1-2x) = 3(1-2x)^2 + 10$
- $(g \circ f)(x) = g(f(x)) = g\left(3x^2+10\right) = 1 - 2\left(3x^2+10\right)$
- $(g \circ g)(x) = g(g(x)) = g(1-2x) = 1 - 2\left(1-2x\right)$

To create a composition function of two user functions at a certain input value, you can prompt the user to enter the functions as strings and dynamically evaluate and compose them.

For $(f \circ g)(x)$ from above example:

```
Enter the first function: x**2
Enter the second function: 3*x+1
Enter the input value: 3
Result: 100.0
```

Inverse Functions

Given a one-to-one function, a new function can sometimes be created by reversing the input and output of the original function. This reversed function is called an **inverse function**.

The inverse function of a one-to-one function $f(x)$ is denoted $f^{-1}(x)$.

Composing a function and its inverse will undo what each of the functions do to the input x, hence $\left(f \circ f^{-1}\right)(x) = x$ and $\left(f^{-1} \circ f\right)(x) = x$.

Example 5

Given $f(x) = 2x + 3$. What f does here is first multiply x by 2, then adding the result by 3.

To reverse what f does, we took the reverse of the action backward. The reverse of "add 3" is "subtract 3". Reverse of "multiply by 2" is "divide by 2".

This means the inverse function first subtracts the input x by 3, then divides the result by 2. Hence the inverse function is $f^{-1}(x) = \frac{x-3}{2}$.

This can easily checked by confirming that $\left(f \circ f^{-1}\right)(x) = x$ and $\left(f^{-1} \circ f\right)(x) = x$.

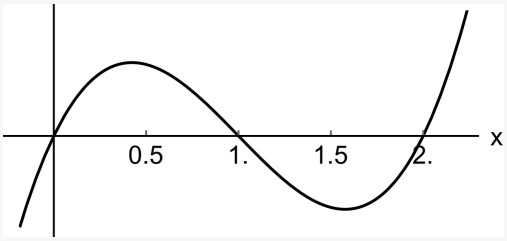
Below is a code example that prompts the user to input a function $f(x)$ and $x = a$, and then calculates and returns $f^{-1}(a)$. Note that the user must input a one-to-one function.

```
Enter the function f(x): x**3
Enter a value for x: 4
Inverse of f(x) at x = 4.0 is approximately: 1.5874
```

8.2.5 Exercises

Exercises

1. Find the domain of the following functions.
   - $f(x) = \sqrt{x - 9}$
   - $f(x) = \dfrac{1}{\sqrt{x} - 9}$
   - $f(x) = \ln(x - 9)$
2. Given the graph of the function f below.



- Find the intervals on which the graph of f is increasing or decreasing
- Find the intervals on which the graph of f is concave up or concave down
- Find the inflection points (if any)

3. Plot the following functions on the interval $[-5, 5]$:
   - $f(x) = 4 + 3x$
   - $f(x) = 4 \cdot 3^x$
   - $f(x) = 4 + 3\log(x)$
   - $f(x) = 4\cos(3x)$
   - $f(x) = \dfrac{1.5}{1 + 4e^{-2.45x}}$
4. Given $f(x) = 1 + 6x^2$ and $g(x) = 6 - \sqrt{x}$. Find each of the following:
   - $[f + g](1234)$
   - $(g \circ f)(2345)$
   - $[f \cdot g](0.3456)$
   - $(f \circ f)(4567)$
   - $g^{-1}(0.2345)$

8.3 Limits, Continuity, and Rates

Note

Below is the list of topics that are covered in this section:

- Limits
- Continuity
- Measures of Change over an Interval
- Measures of Change at a Point
- Secant and Tangent Lines

Note

Please visit the web version of this section to engage with this interactive resource.

```
import sympy as sp
from scipy.optimize import fsolve
import numpy as np
import matplotlib.pyplot as plt
import math
```

8.3.1 Limits

In the previous section, while talking about behaviors of some common functions, we saw the phrase "$f(x)$ goes to ∞ as x goes to ∞".

This phrase describes how the output of a function behaves as the input is **approaching** something and can be written using **limit** notation: $\lim_{x\to\infty} f(x) = \infty$.

The keyword here is "approaching". The input is not equal to a certain point, it is approaching it.

We say that the **limit** of $f(x)$ is L as x approaches a and write this as $\lim_{x\to a} f(x) = L$ provided we can make $f(x)$ as close to L as needed for all x sufficiently close to a, from both sides, without actually letting x be a.

$\lim_{x\to a} f(x) = L$ can also alternatively be written as $f(x) \to L$ as $x \to a$.

The phrase "from both sides" refer to the **Left-hand limits** and **Right-hand limits**.

For function f defined on an interval containing a constant a (except possibly at a itself):

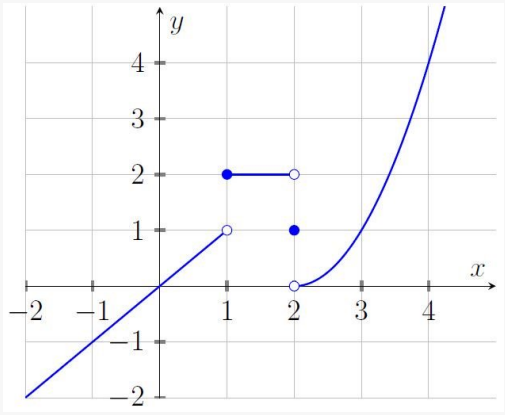
- Left-hand limit of f as x approaches a from the left is written $\lim_{x\to a^-} f(x) = L_1$
- Right-hand limit of f as x approaches a from the right is written $\lim_{x\to a^+} f(x) = L_2$

Here a^- denotes "to the left of a and a^+ denotes "to the right of a.

By definition, for function f defined on an interval containing a constant a (except possibly at a itself), the limit of f as x approaches a exists if and only if $\lim_{x\to a^-} f(x) = \lim_{x\to a^+} f(x) = L$, and hence written $\lim_{x\to a} f(x) = L$

Example 1

Given the graph of $f(x)$ below:



- $\lim_{x\to 0^-} f(x) = 0$ while $\lim_{x\to 0^+} f(x) = 0$ hence $\lim_{x\to 0} f(x) = 0$.

- $\lim_{x \to 1^-} f(x) = 1$ while $\lim_{x \to 1^+} f(x) = 2$ hence $\lim_{x \to 1} f(x)$ does not exists.

Note that when looking for limits as x approaching 0 or 1, it never mattered what $f(0)$ or $f(1)$ is, as we only see the values to the left and right of those points.

Below is a code example that prompt user for a function $f(x)$ and a constant a, and then return $\lim_{x \to a} f(x) = L$.

```
The limit of f(x) as x approaches 3.8 is: -2.44743156377088
```

Limit Properties

Assuming $\lim_{x \to a} f(x)$ and $\lim_{x \to a} g(x)$ exists and c is any constant. Then:

- $\lim_{x \to a} [c \cdot f(x)] = c \cdot \lim_{x \to a} f(x)$
- $\lim_{x \to a} [f(x) \pm g(x)] = \lim_{x \to a} f(x) \pm \lim_{x \to a} g(x)$
- $\lim_{x \to a} [f(x) \cdot g(x)] = \lim_{x \to a} f(x) \cdot \lim_{x \to a} g(x)$
- $\lim_{x \to a} \left[\dfrac{f(x)}{g(x)}\right] = \dfrac{\lim_{x \to a} f(x)}{\lim_{x \to a} g(x)}$, provided $\lim_{x \to a} g(x) \neq 0$
- $\lim_{x \to a} [f(x)]^n = \left[\lim_{x \to a} f(x)\right]^n$
- $\lim_{x \to a} c = c$
- $\lim_{x \to a} x = a$

8.3.2 Continuity

Graphically, a function is **continuous** if it can be drawn without lifting the writing utensil from the paper.

Mathematically, a function f is **continuous at** $x = a$ if and only if the following three conditions are satisfied:

- $f(a)$ exists
- $\lim_{x \to a} f(x)$ exists
- $\lim_{x \to a} f(x) = f(a)$

A function is **continuous on an interval** if these three conditions are met for every input value in the interval.

Fact: If $\lim_{x \to a} g(x) = b$ and $f(x)$ is continuous at $x = b$, then $\lim_{x \to a} f(g(x)) = f\left(\lim_{x \to a} g(x)\right)$.

Example

Evaluate $\lim_{x \to 0} e^{\cos(x)}$.

The function $\cos(x)$ is continuous at $x = 0$ since $\cos(0) = \lim_{x \to 0^-} \cos(x) = \lim_{x \to 0^+} \cos(x) = 1$.

Hence $\lim_{x \to 0} e^{\cos(x)} = e^{\left(\lim_{x \to 0} \cos(x)\right)} = e^1 = e$. Of course, this can easily be checked using the code.

```
The limit of e^(cos(x)) as x approaches 0 is: E
```

Intermediate Value Theorem

Suppose that $f(x)$ is continuous on $[a, b]$ and let M be any number between $f(a)$ and $f(b)$. Then there exists a constant c such that $a < c < b$ and $f(c) = M$.

Example

Show that the function $f(x) = x^3 + 2x^2 - 0.3$ has a zero in the interval $[0, 1]$.

Since $f(x)$ is a polynomial, it is continuous everywhere. $f(0) = -0.3$ and $f(1) = 2.7$.

Since 0 is between -0.3 and 2.7, then by the Intermediate Value Theorem, there is a constant c between 0 and 1 such that $f(c) = 0$, that is, c is a zero of $f(x)$ on $[0, 1]$.

This theorem can be useful for computationally expensive functions. Otherwise, with the help of any calculating tools, it is quite simple to just find all the zeros of a given function in a given interval. Below is a code example that prompts the user for $f(x)$ and interval $[a, b]$, and then returns the zeroes located in $[a, b]$.

```
Zeros of the function within the interval [a, b]: {-0.438, 0.357, -1.918}
```

8.3.3 Measures of Change over an Interval

If x_1 and x_2 are input values for the function f and $x_1 < x_2$, then:

- the **change** in f from x_1 to x_2 is $f(x_2) - f(x_1)$
  - what it calculates: between input x_1 and x_2, by how much does the output changes?
- the **percent change** in f from x_1 to x_2 is $\dfrac{f(x_2) - f(x_1)}{f(x_1)} \cdot 100\%$
  - what it calculates: between input x_1 and x_2, by how many % does the output changes?
- the **average rate of change (aRoC)** in f from x_1 to x_2 is $\dfrac{f(x_2) - f(x_1)}{x_2 - x_1}$
  - what it calculates: between input x_1 and x_2, how fast does the output changes?

Below is an example of code that prompt user to input two points $(x_1, f(x_1))$ and $(x_2, f(x_2))$, then return the change, percent change, and average rate of change between the two points.

```
Change in y: 30.0
Percent change: 150.0 %
Average rate of change: 15.0
```

8.3.4 Measures of Change at a Point

If x is an input values for the function f, then:

- the **(instantaneous) rate of change (RoC)** of f at point $x = a$ is $\lim_{x \to a} \dfrac{f(x) - f(a)}{x - a}$
  - what it calculates: at the input $x = a$, how fast does the output changes?
- the **percent rate of change** of f at point $x = a$ is $\dfrac{\text{RoC}}{f(a)} \cdot 100\%$
  - what it calculates: at the input $x = a$, how fast does the output percentage changes?

Below is an example of code that prompts the user to input a function $f(x)$ and the point $x = a$, then return the instantaneous rate of change and percent rate of change of f at $x = a$.

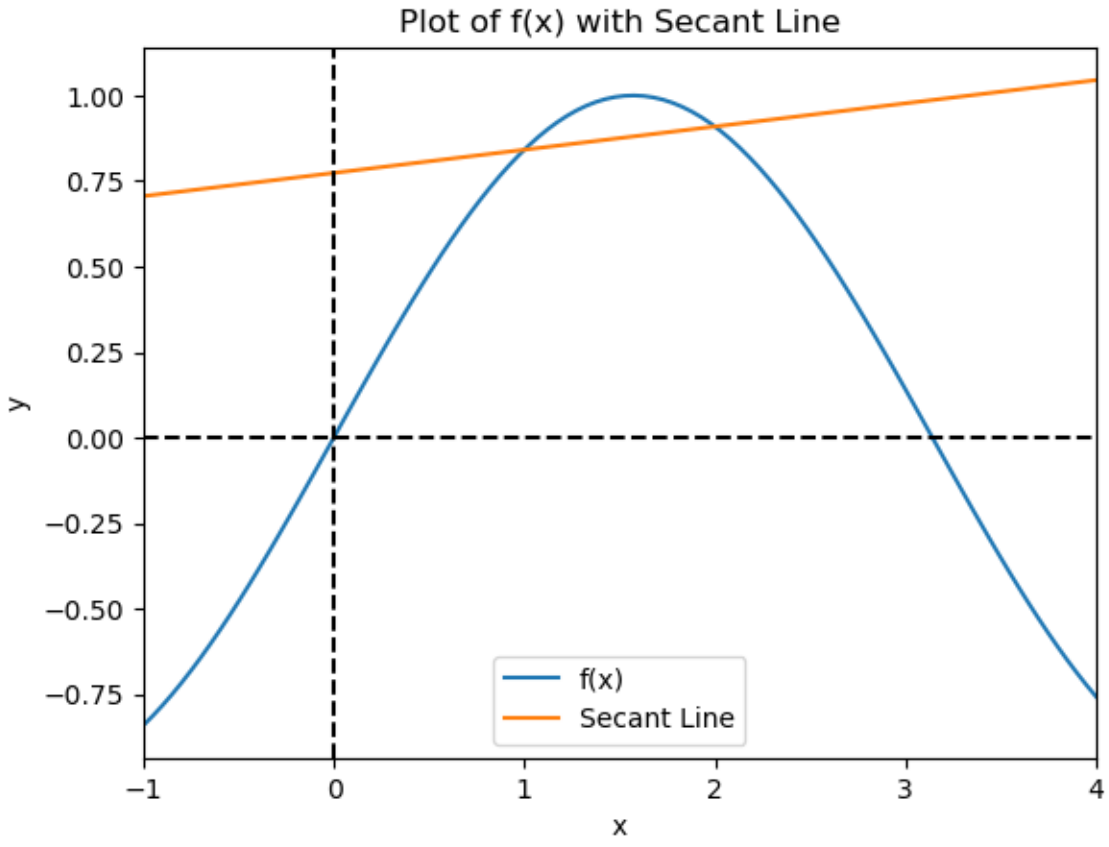
```
Approximate Instantaneous Rate of Change: 28.0
Approximate Percent Rate of Change: 38.889
```

8.3.5 Secant and Tangent Lines

Given a continuous function $f(x)$ on the interval containing $x = a$ and $x = b$, the **secant line** on f between $x = a$ and $x = b$ is a straight line connecting $(a, f(a))$ and $(b, f(b))$.

The slope of the secant line is the average rate of change of f between $x = a$ and $x = b$.

Below is an example code prompting the user to input function f as well as $x = a$ and $x = b$. Code will return the plot of f together with its secant line between $x = a$ and $x = b$.

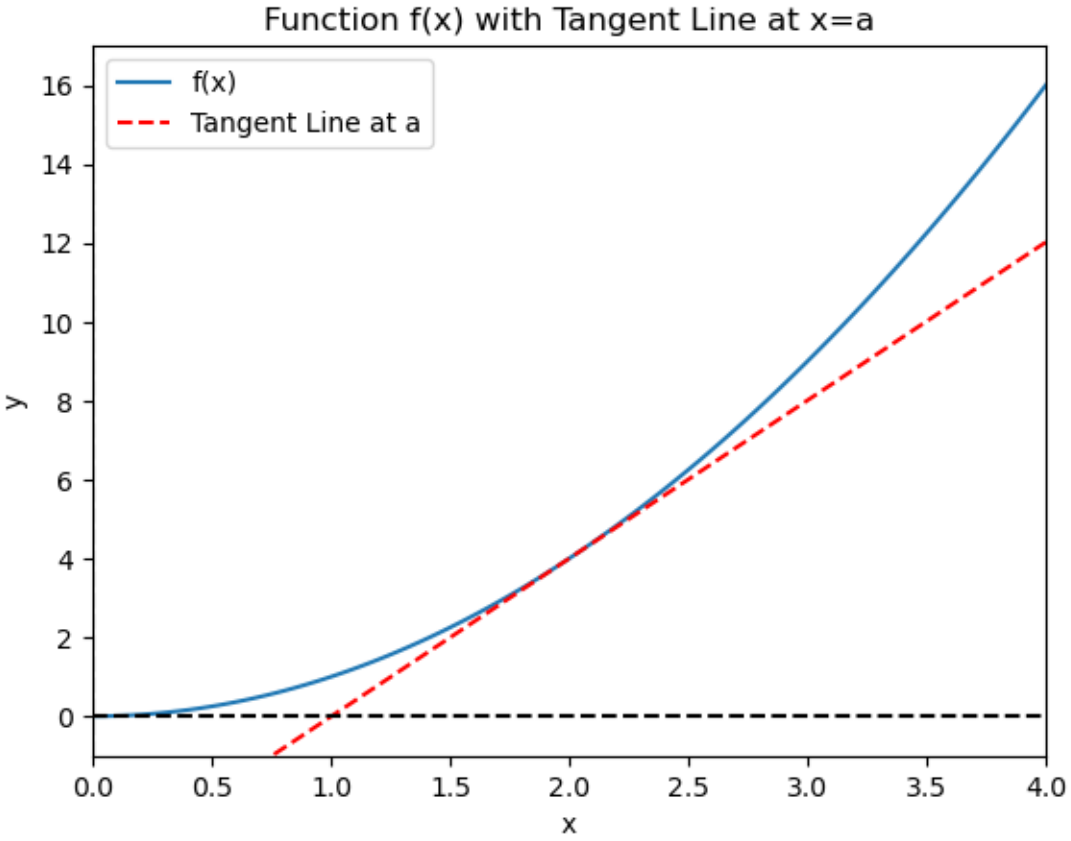


The word **tangent** comes from the Latin word *tangere*, meaning "to touch".

Given a continuous function $f(x)$ on the interval containing $x = a$, the **tangent line** on f at $x = a$ is a straight line touching $f(x)$ at exactly $(a, f(a))$ and nowhere else.

The slope of this tangent line is the instantaneous rate of change of f at $x = a$.

Below is an example code prompting user to input function f and $x = a$. Code will return the plot of f together with its tangent line at $x = a$.

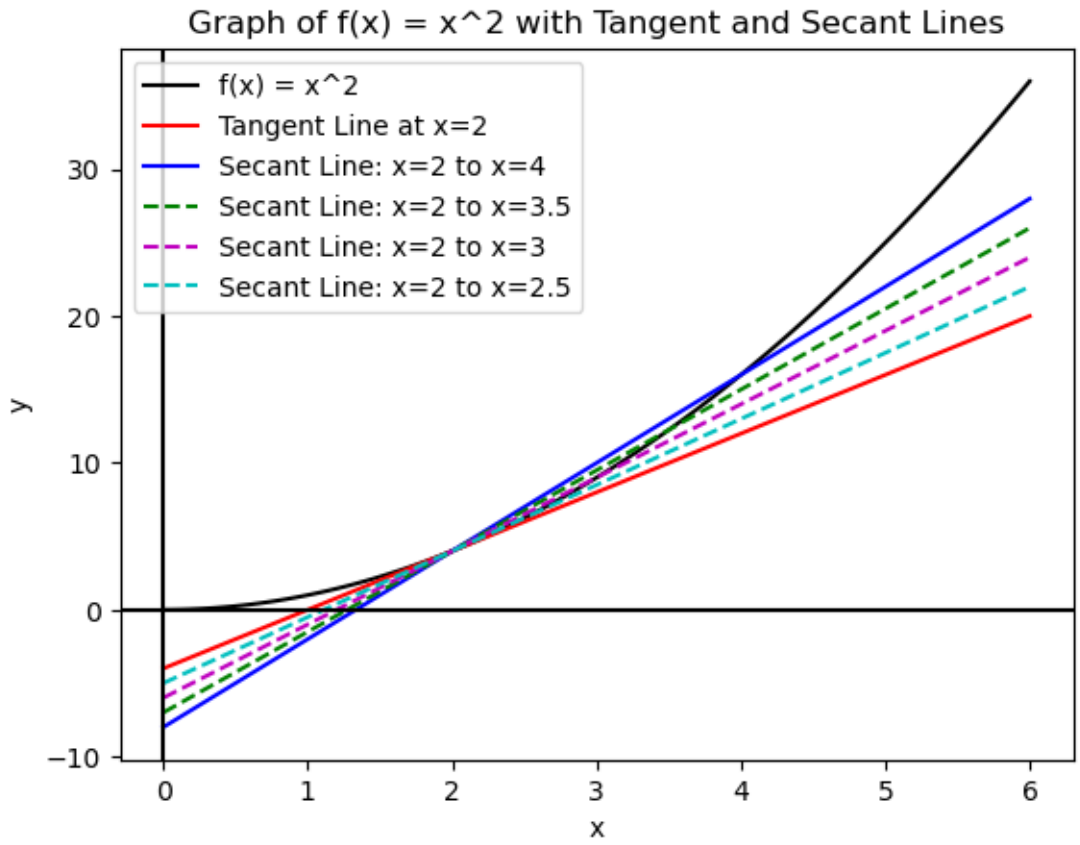


The tangent line at $x = a$ can be obtained by taking the secant line between $x = a$ and $x = b$, then bring b closer and closer to a.

Taking $b = a + h$, the slope of the secant line between $x = a$ and $x = b$ is $\dfrac{f(a+h) - f(a)}{a + h - a} = \dfrac{f(a+h) - f(a)}{h}$.

Bringing $b \to a$ is then bringing $h \to 0$, and hence the slope of the tangent line at $x = a$ is $\lim\limits_{h \to 0} \dfrac{f(a+h) - f(a)}{h}$. This realization is important as we introduce derivatives in the next section.

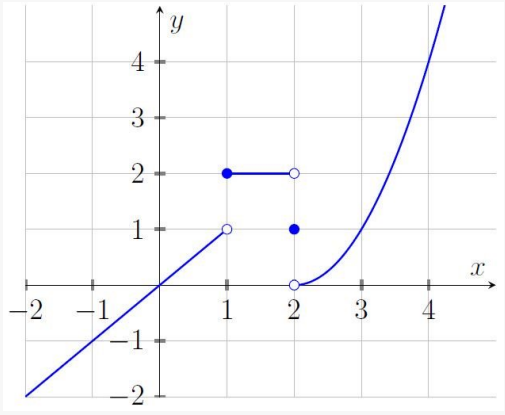
Below is the code showing the graph of $f(x) = x^2$ with the secant lines between $x = 2$ and $x = b$, starting from $b = 4$ until $b = 2$, which gives the tangent line of $f(x)$ at $x = 2$.



8.3.6 Exercises

Exercises

1. Given the graph of $f(x)$ below:



Determine whether or not $\lim_{x\to 2} f(x)$ and $\lim_{x\to 3} f(x)$ exists. Explain.

2. Calculate the following limits. Explain why or why not the limit exists.

   - $\lim_{x\to 7} 5\ln(x)$

- $\lim_{x \to 3} \left(3x^3 + 4x^2 - 5x - 6\right)$
- $\lim_{x \to 0} \frac{1}{x}$

3. Do the following problems:
   - Evaluate $\lim_{x \to 0} e^{\sin(x)}$
   - Find all zeroes of $f(x) = 4x^3 + 3x^2 - 3x + 0.2$ on the interval $[0, 1]$
4. Given $f(x) = 4x^3 - 2x^2 + 3x - 7$. Calculate the following:
   - Change of f between $x = -1$ and $x = 2$
   - Percent change of f between $x = -1$ and $x = 2$
   - Average rate of change of f between $x = -1$ and $x = 2$
   - Instantaneous rate of change of f at $x = 3$
   - Percent rate of change of f at $x = 3$
5. Given $f(x) = 4x^3 - 2x^2 + 3x - 7$. Plot the following:
   - Secant line between on f between $x = -1$ and $x = 2$
   - Tangent line on f at $x = 3$

8.4 Derivatives

Note

Below is the list of topics that are covered in this section:

- Derivatives and Derivative Notations
- Properties, Formulas, and Rules
- Derivative Graphs
- Extrema
- The Mean Value Theorem
- Linearization
- Implicit Differentiation

```
import sympy as sp
import numpy as np
import matplotlib.pyplot as plt
```

8.4.1 Derivatives and Derivative Notations

Derivative shows the sensitivity of change of a function's output when the input changes. From the previous section, this is the same as what is being measured by the instantaneous rate of change (RoC), which is the slope of the tangent line at a certain point.

One notation of the derivative of the function $f(x)$ is $f'(x)$, read "f prime of x". Hence, the derivative of the function $f(x)$ at the point $x = a$ is $f'(a) = \lim_{h \to 0} \dfrac{f(a+h) - f(a)}{h}$.

For the function $f(x)$, the **derivative of** $f(x)$ **with respect to** x, or, some say the **limit definition of the derivative of** $f(x)$, is defined as

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

If derivative $f'(x)$ exists at all points in the interval (a, b), then $f(x)$ is said to be **differentiable** in (a, b).

Example

Find the derivative of $f(x) = 5x^2 - 2x$ using the definition of the derivative.

$$\begin{aligned}
f'(x) &= \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} && (8.1)\\
&= \lim_{h \to 0} \frac{[5(x+h)^2 - 2(x+h)] - [5x^2 - 2x]}{h} && (8.2)\\
&= \lim_{h \to 0} \frac{[5\left(x^2 + 2xh + h^2\right) - 2(x+h)] - [5x^2 - 2x]}{h} && (8.3)\\
&= \lim_{h \to 0} \frac{5x^2 + 10xh + 5h^2 - 2x - 2h - 5x^2 + 2x}{h} && (8.4)\\
&= \lim_{h \to 0} \frac{10xh + 5h^2 - 2h}{h} && (8.5)\\
&= \lim_{h \to 0} \frac{h\left(10x + 5h - 2\right)}{h} && (8.6)\\
&= \lim_{h \to 0} \left(10x + 5h - 2\right) && (8.7)\\
&= 10x + 5(0) - 2 = 10x && (8.8)
\end{aligned}$$

The notation $f'(x)$ is the **Lagrange's** notation for derivative. The derivative of $f'(x)$, or the second derivative of $f(x)$ is denoted by $f''(x)$ or $f^{(2)}(x)$. The derivative of $f^{(2)}(x)$, or the third derivative of $f(x)$ is denoted by $f^{(3)}(x)$, and so on.

Various notations for derivatives have been proposed by various mathematicians, and some are used independently in different locations and/or eras. Some common notations for derivatives are given below.

- Lagrange: $f'(x), f^{(2)}(x), f^{(3)}(x), \ldots$
- Leibniz: $\dfrac{df}{dx}, \dfrac{d^2f}{dx^2}, \dfrac{d^3f}{dx^3}, \ldots$
- Euler: $\left(Df\right)(x), \left(D^2f\right)(x), \left(D^3f\right)(x), \ldots$
- Newton: $\dot{f}, \ddot{f}, \dddot{f}, \ldots$

Motivated students are encouraged to browse the variations of these notations for multivariable functions or higher-level mathematics.

Lagrange's and **Leibniz's** notation are two of the most used derivative notations, at least restricted to undergraduate Calculus courses.

8.4.2 Properties, Formulas, and Rules

Derivative Properties

- Derivative of sum (or difference) is the sum (or difference) of derivative.

  $[f \pm g]' = f' \pm g' \qquad \text{or} \qquad \frac{d}{dx}[f \pm g] = \frac{df}{dx} \pm \frac{dg}{dx}$

- Constant multiple can be factored outside derivative.

  $[c \cdot f]' = c \cdot f' \qquad \text{or} \qquad \frac{d}{dx}[c \cdot f] = c \cdot \frac{df}{dx}$

Both properties can be proved using the limit definition of derivative and some basic algebraic manipulation. Below is the proof of the second property, while the proof of the first property is one of the exercises for the reader.

Proof of $[c \cdot f]' = c \cdot f'$

$$[c \cdot f]' = \lim_{h \to 0} \frac{cf(x+h) - cf(x)}{h} = \lim_{h \to 0} c \cdot \frac{f(x+h) - f(x)}{h} = c \cdot \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} = c \cdot f'$$

Basic Derivative Formulas

While limit definitions of derivatives can always be used to find the derivative of a function, here are some common derivative formulas that will be helpful to remember to save time and lower the risk of algebraic manipulations. Motivated students are encouraged to try proving these formulas.

Derivative of:

- Constant by itself: $\frac{d}{dx}[c] = 0$
- Power function: $\frac{d}{dx}[x^n] = nx^{n-1}$
- Exponential function: $\frac{d}{dx}[b^x] = b^x \cdot \ln(b)$
- Special exponential function when $b = e$: $\frac{d}{dx}[e^x] = e^x \cdot \ln(e) = e^x \cdot 1 = e^x$
- Natural log function: $\frac{d}{dx}[\ln(x)] = \frac{1}{x}$
- Sine function: $\frac{d}{dx}[\sin(x)] = \cos(x)$
- Cosine function: $\frac{d}{dx}[\cos(x)] = -\sin(x)$

Product, Quotient, and Chain Rule

The limit definition of derivatives can also be used to prove the derivative of product, quotient, and composition of two functions:

- Product Rule:

  $[f \cdot g]' = f' \cdot g + f \cdot g'$

- Quotient Rule:

  $\left[\frac{f}{g}\right]' = \frac{f' \cdot g - f \cdot g'}{g^2}$

- Chain Rule - for composition of two functions $(f \circ g)(x) = f(g(x))$:

  $[(f \circ g)(x)]' = f'(g(x)) \cdot g'(x)$ or $\frac{df}{dx} = \frac{df}{dg} \cdot \frac{dg}{dx}$

Derivative Rules - Examples and Code

- Find the derivative of $a(x) = x^3 \cdot 3^x$.
  - Note that $a(x)$ is a product $f(x) \cdot g(x)$ with $f(x) = x^3$ and $g(x) = 3^x$. Using basic derivative formulas, we have $f'(x) = 3x^2$ and $g'(x) = 3^x \ln(3)$. Using Product Rule, we have $a'(x) = 3x^2 \cdot 3^x + x^3 \cdot 3^x \ln(3)$.
- Find the derivative of $b(x) = \dfrac{3x+1}{\sqrt{x}+2}$.
  - Note that $b(x)$ is a quotient $\dfrac{f(x)}{g(x)}$ with $f(x) = 3x + 1$ and $g(x) = \sqrt{x} + 2 = x^{1/2} + 2$. Using basic derivative formulas, we have $f'(x) = 3$ and $g'(x) = \dfrac{1}{2}x^{-1/2}$. Using Quotient Rule, we have $b'(x) = \dfrac{3 \cdot \left(x^{1/2}+2\right) - (3x+1) \cdot \left(\frac{1}{2}x^{-1/2}\right)}{\left(x^{1/2}+2\right)^2}$.
  - Note that $b(x) = \dfrac{3x+1}{\sqrt{x}+2}$ can be written as the product of two functions: $b(x) = (3x+1)\left(\sqrt{x}+2\right)^{-1}$. Hence, $b'(x)$ can also be found by using product rule. Here $f(x) = (3x+1)$ so $f'(x) = 3$. While $g(x) = \left(x^{1/2}+2\right)^{-1}$ so applying Chain Rule we have $g'(x) = -\left(x^{1/2}+2\right)^{-2} \cdot \frac{1}{2}x^{-1/2}$. Using Product Rule, we have $b'(x) = 3 \cdot \left(x^{1/2}+2\right)^{-1} + (3x+1) \cdot \left(-\left(x^{1/2}+2\right)^{-2}\right) \cdot \frac{1}{2}x^{-1/2}$. It is quite an easy exercise to see that this $b'(x)$ is equivalent to $b'(x)$ obtained by using the Quotient Rule above.
- Find the derivative of $c(x) = \sin\left(2x^3 + \ln(x)\right)$.
  - Note that $c(x)$ is a composition function $(f \circ g)(x)$ with $f(x) = \sin(x)$ and $g(x) = 2x^3 + \ln(x)$. Using basic derivative formulas, we have $f'(x) = \cos(x)$ and $g'(x) = 6x + \frac{1}{x}$. Using Quotient Rule, we have $c'(x) = \cos\left(2x^3 + \ln(x)\right) \cdot \left(6x + \frac{1}{x}\right)$.

Below is an example code that prompts the user to input the function $f(x)$ and return the derivative function $f'(x)$. Note that Python uses **log** instead of **ln** for natural log. Also, the user must input and read the exponent and multiplication symbols carefully.

```
# Prompt user for the function f(x)
f_input = input("Enter the function f(x): ")

# Define the variable and the function symbolically
x = sp.Symbol('x')
```

(continues on next page)

(continued from previous page)

```
f = sp.sympify(f_input)

# Calculate the derivative
f_prime = sp.diff(f, x)

# Print the derivative
print("The derivative f'(x) is:", f_prime)
```

```
The derivative f'(x) is: 3**x*x**3*log(3) + 3*3**x*x**2
```

L'Hospital's Rule and Indeterminate Forms

In the previous section, we discussed limits. Recall that there are cases where the limit of $f(x)$ exists at $x = a$ since the left and right limits are equal, but $f(a)$ itself does not exist.

Algebraically, we can use the following examples: $\lim\limits_{x\to 2} \dfrac{x^2-4}{x-2}$ and $\lim\limits_{x\to\infty} \dfrac{3x^2-2x}{5x^2+7x}$.

If we apply the $x \to 2$ and $x \to \infty$ directly into these limits, the first limit gives $\frac{0}{0}$ while the second limit gives $\frac{\infty}{\infty}$. However, with some algebraic manipulation, we can calculate the limits:

- $\lim\limits_{x\to 2} \dfrac{x^2-4}{x-2} = \lim\limits_{x\to 2} \dfrac{(x+2)(x-2)}{x-2} = \lim\limits_{x\to 2} (x+2) = 4$
- $\lim\limits_{x\to\infty} \dfrac{3x^2-2x}{5x^2+7x} = \lim\limits_{x\to\infty} \dfrac{3-\frac{2}{x}}{5+\frac{7}{x}} = \dfrac{3}{5}$

Both $\frac{0}{0}$ and $\frac{\infty}{\infty}$ are **indeterminate forms**. Some other indeterminate forms are $0\cdot\infty, 1^\infty, 0^0, \infty^0, \infty-\infty$. These forms have conflicting behaviors and it is not clear which behavior dominates another hence the result is indeterminate.

L'Hospital's Rule is a rule that utilizes derivatives to calculate limits of indeterminate forms $\frac{0}{0}$ or $\frac{\infty}{\infty}$, that are often simpler than doing algebraic manipulation: If $\lim\limits_{x\to a} \dfrac{f(x)}{g(x)} = \dfrac{0}{0}$ or $\lim\limits_{x\to a} \dfrac{f(x)}{g(x)} = \pm\dfrac{\infty}{\infty}$, where a can be any real number or $\pm\infty$, then $\lim\limits_{x\to a} \dfrac{f(x)}{g(x)} = \lim\limits_{x\to a} \dfrac{f'(x)}{g'(x)}$.

Above examples that use L'Hospital's Rule:

- $\lim\limits_{x\to 2} \dfrac{x^2-4}{x-2} = \lim\limits_{x\to 2} \dfrac{2x}{1} = \lim\limits_{x\to 2} (2x) = 4$
- $\lim\limits_{x\to\infty} \dfrac{3x^2-2x}{5x^2+7x} = \lim\limits_{x\to\infty} \dfrac{6x-2}{10x+7} = \lim\limits_{x\to\infty} \dfrac{6}{10} = \dfrac{3}{5}$

Note that with the code given on the previous section, it does not matter if the limit is of indeterminate form $\frac{0}{0}$ or $\frac{\infty}{\infty}$.

```
# Prompt the user to input the function f(x) and the constant a
expression_str = input("Enter the function f(x): ")
a = float(input("Enter the constant a: "))

# Define the variable x and the function f(x)
x = sp.symbols('x')
f_x = sp.sympify(expression_str)

# Calculate the limit of f(x) as x approaches a
limit_result = sp.limit(f_x, x, a)
```

(continues on next page)

(continued from previous page)

```
# Print the result
print("The limit of f(x) as x approaches", a, "is:", limit_result)
```

```
The limit of f(x) as x approaches 2.0 is: 4.00000000000000
```

8.4.3 Derivative Graphs

Increasing and Decreasing

Given any inputs x_1 and x_2 from an interval I with $x_1 < x_2$.

- If $f(x_1) < f(x_2)$, then $f(x)$ is **increasing** on I.
- If $f(x_1) > f(x_2)$, then $f(x)$ is **decreasing** on I.

Since $f'(a)$ is the slope of the tangent line of $f(x)$ at $x = a$, then if for every input x on an interval I:

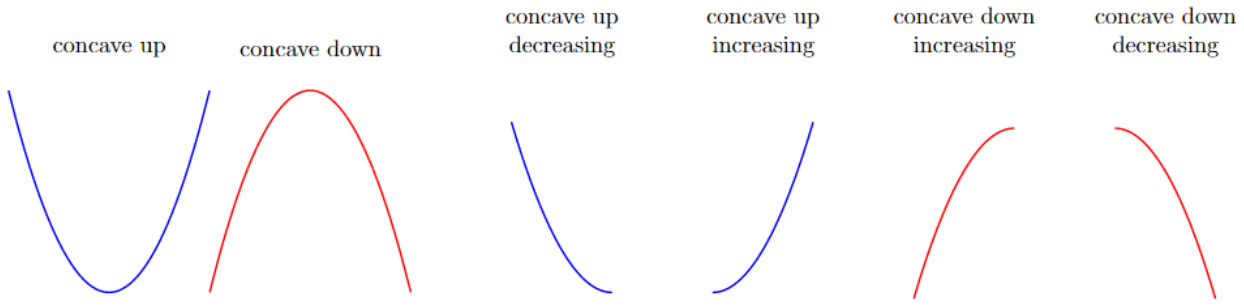
- $f'(x) > 0 \Rightarrow$ slope of tangent line is positive $\Rightarrow$ tangent line is increasing $\Rightarrow f(x)$ is increasing.
- $f'(x) < 0 \Rightarrow$ slope of tangent line is negative $\Rightarrow$ tangent line is decreasing $\Rightarrow f(x)$ is decreasing.
- $f'(x) = 0 \Rightarrow$ slope of tangent line is zero $\Rightarrow$ tangent line is flat $\Rightarrow f(x)$ is constant.

Below is an example code that asks the user for the function $f(x)$ and the interval $I = (a, b)$. It will return the intervals within (a, b) where $f(x)$ increases or decreases.

```
Interval (-4.0, -2.0): Decreasing
Interval (-2.0, 4.0): Increasing
Interval (4.0, 5.0): Decreasing
```

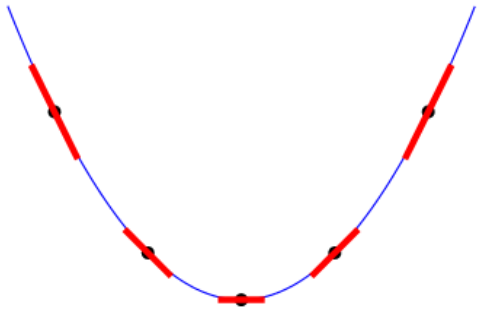
Concave Up and Concave Down

A function is **concave up** if it "opens" upward. A function is **concave down** if it "opens" downward.



A point $x = c$ is called an **inflection point** of $f(x)$ if the graph of $f(x)$ is continuous and its concavity changes at $x = c$.

Suppose the graph of $f(x)$ is concave up as given below. Draw tangent lines across the inputs of $f(x)$.



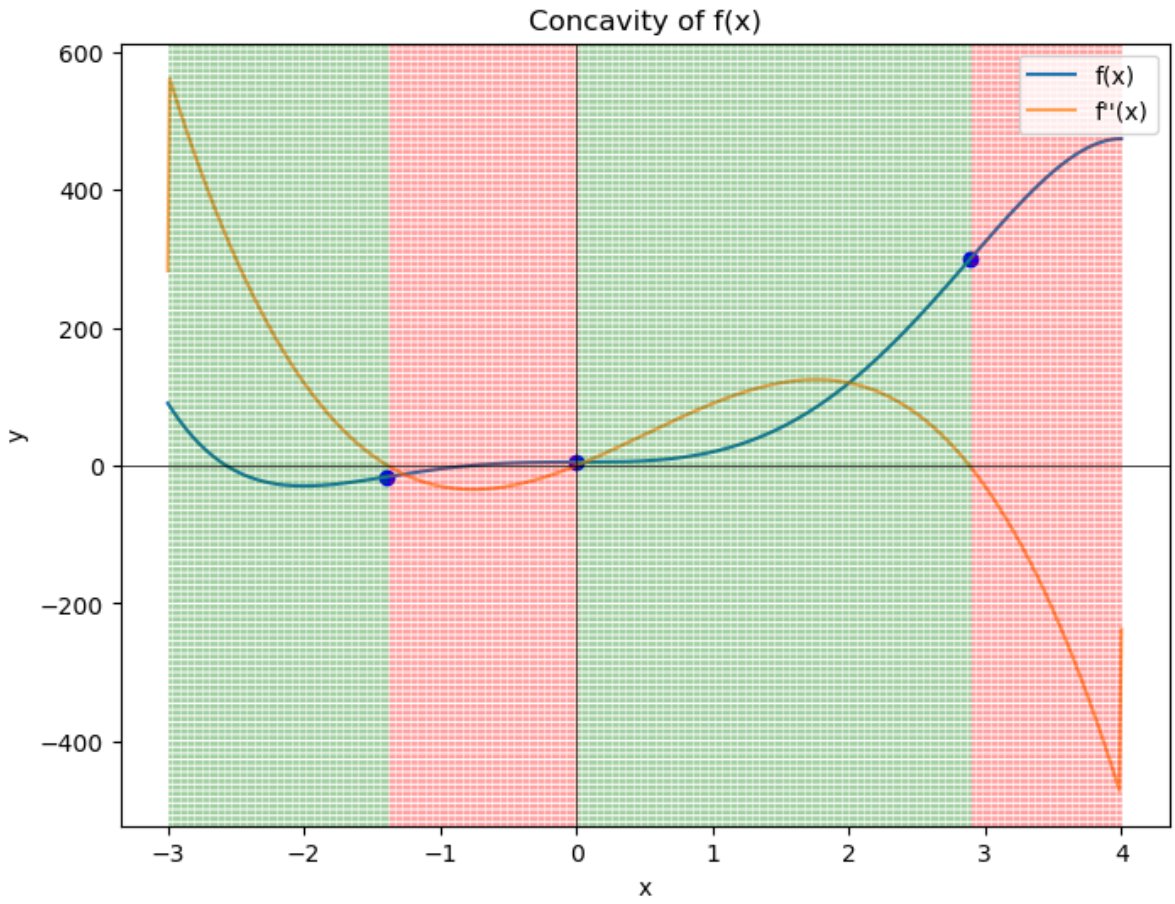
Note that as the value of a increases, the slope of the tangent line of $f(x)$ at $x = a$ also increases (from negative, to zero, to positive). This implies that if the graph of $f'(x)$ is increasing, then the graph of $f(x)$ is concave up. From the previous subsection about Increasing and Decreasing, if the graph of $f''(x)$ is positive then the graph of $f'(x)$ is increasing.

Apply a similar argument for cases where the graph of $f(x)$ is concave down, we can summarize that if for all x in some interval I we have:

- $f''(x) > 0$ then $f(x)$ is concave up on I.
- $f''(x) < 0$ then $f(x)$ is concave down on I.

Note that if $f''(x) = 0$ then we cannot guarantee the shape of the graph of $f(x)$. $f''(x) = 0$ only means the graph of $f'(x)$ is flat, and many different cases that can cause this. Can you think of these cases?

Below is an example code that asks the user for the function $f(x)$ and the interval $I = (a, b)$. It will plot the graph of $f(x)$ and $f''(x)$ on (a, b), list all the inflection points, and highlight the area on which $f(x)$ concave up (or concave down) with green (or red).




```
Inflection Points:
(-1.388, -16.236)
(-0.001, 5.000)
(2.886, 298.696)
```

8.4.4 Extrema

Many applications of Calculus revolve around maximizing or minimizing values of a function. While visualizing the extrema can be done by analyzing the graph of the function, it might be computationally cheaper to find the extrema using derivatives.

Definition

Relative Extrema

- $f(x)$ has a **relative (or local) maximum** at $x = a$ if $f(x) > f(a)$ for every x in some open interval around $x = a$ (in the "local" neighborhood of $x = a$).
- $f(x)$ has a **relative (or local) minimum** at $x = a$ if $f(x) < f(a)$ for every x in some open interval around $x = a$ (in the "local" neighborhood of $x = a$).

Relative extrema cannot happen at the endpoints of the domain (since the open interval around $x = a$ should include some value both to the left and right of $x = a$).

Absolute Extrema

- $f(x)$ has a **absolute (or global) maximum** at $x = a$ if $f(x) \leq f(a)$ for every x in the domain of f.
- $f(x)$ has a **absolute (or global) minimum** at $x = a$ if $f(x) \geq f(a)$ for every x in the domain of f.

Extreme Value Theorem

Theorem: Suppose that $f(x)$ is continuous on the interval $[a, b]$ then there are two numbers c, d where $a \leq c$ and $d \leq b$ such that $f(c)$ is an absolute maximum for f and $f(d)$ is an absolute minimum of f.

Note that this theorem doesn't provide the information on where the absolute extrema are, or whether they will occur more than once. It only says that absolute extrema exists somewhere in the interval.

On the other hand, since relative extrema cannot occur at the endpoints, the existence of relative extrema is not guaranteed.

One easy example is the constant function $f(x) = k$ for some real number k. This function has no relative extrema in any interval, but the absolute max and absolute min are both k (that occurs infinitely many times at every point in the domain).

Finding Extrema

A **critical number** of a function $f(x)$ is $x = c$ where $f(c)$ exists, but either $f'(c) = 0$ or $f'(c)$ does not exist. The point $(c, f(c))$ is called a **critical point**.

- $f'(c) = 0$ means the tangent line of $f(x)$ at $x = c$ has slope 0, meaning it is a horizontal tangent line. This can happen on the relative max, relative min, or an inflection point(concave up or down).
- $f'(c)$ does not exist, which means there is no tangent line, or a tangent line exists but is vertical (has an undefined slope). This can happen at a corner/cusp when the function is not continuous, or at an inflection point.

Using critical numbers, we have the First Derivative Test:

First Derivative Test
Suppose that $x = c$ is a critical point of $f(x)$, then:

- If $f'(x)$ change sign from $+$ to $-$ at $x = c$, then $f(x)$ has a relative maximum at $x = c$.
- If $f'(x)$ change sign from $-$ to $+$ at $x = c$, then $f(x)$ has a relative minimum at $x = c$.
- If $f'(x)$ doesn't change sign at $x = c$, then $f(x)$ has neither a relative maximum nor a relative minimum at $x = c$.

Going further to the second derivative, recall the relation between the second derivative $f''(x)$ and the shape of graph $f(x)$. Note that a concave up graph has a lowest point while a concave down graph has a highest point, which we will use for the Second Derivative Test:

Second Derivative Test
Suppose that $x = c$ is a critical point of $f(x)$ such that $f'(c) = 0$ and that $f''(x)$ is continuous in the local neighborhood around $x = c$, then:

- If $f''(x) < 0$, then $f(x)$ has a relative maximum at $x = c$.
- If $f''(x) > 0$, then $f(x)$ has a relative minimum at $x = c$.
- If $f''(x) = 0$, then at $x = c$ the function $f(x)$ can have a relative maximum, relative minimum, or neither.

Summarizing, below are the steps to find extrema given a continuous function $f(x)$ on the interval $[a, b]$:

- Find the critical points
- Do the first or second derivative test to get the relative extrema
- Evaluate and compare the values of $f(x)$ at the critical points and the endpoints to get the absolute extrema

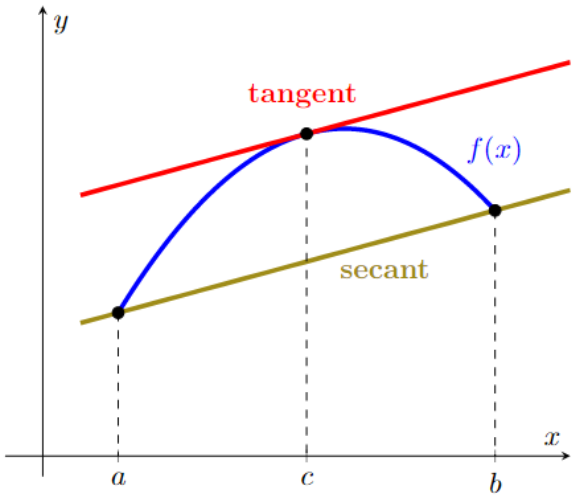
It is left to motivated students to do these steps manually.

Below is the example code that asks the user to give $f(x)$ and the endpoints of closed interval $[a, b]$. Code will return the list of absolute and relative extrema.

```
Relative Maxima:
(4.00000000000000, 474.333)
Relative Minima:
(-2.00000000000000, -29.667)
Absolute Maximum:
(-4.0, 815.667)
Absolute Minimum:
(-2.00000000000000, -29.667)
```

8.4.5 The Mean Value Theorem

Also called the **Lagrange Theorem**, the **Mean Value Theorem (MVT)** stated that, if f is a continuous function on the closed interval $[a, b]$ and differentiable on the open interval (a, b), then there is a point c in (a, b) such that the tangent at c is parallel to the secant line through the endpoints $(a, f(a))$ and $(b, f(b))$, that is, $f'(c) = \dfrac{f(b) - f(a)}{b - a}$, or, $f(b) - f(a) = f'(c)(b - a)$.



Note that MVT doesn't give what c is. It only guarantees the existence of at least one of such c. Finding the c can be done manually or using code.

Example

Determine the number(s) c which satisfies the conclusion of the MVT for the function $f(x) = 4x - x^2$ on $[0.5, 3]$.

The derivative of $f(x)$ is $f'(x) = 4 - 2x$. From the conclusion of the MVT we have $4 - 2c = f'(c) = \dfrac{f(3) - f(0.5)}{3 - 0.5} = \dfrac{1.25}{2.5} = 0.5$, giving $c = 1.75$ as the only such c.

Below is the code that prompts the user to provide $f(x)$ and the interval endpoints a and b, then returns the value(s) of c satisfying the conclusion of MVT.

```
Values of c that satisfy the conclusion of MVT:
c = 1.75000000000000
```

Example

Given $f(x)$ a continuous and differentiable function on $[2, 10]$. It is known that $f(2) = -5$ and $f'(x) \leq 6$. What is the largest possible value for $f(10)$?

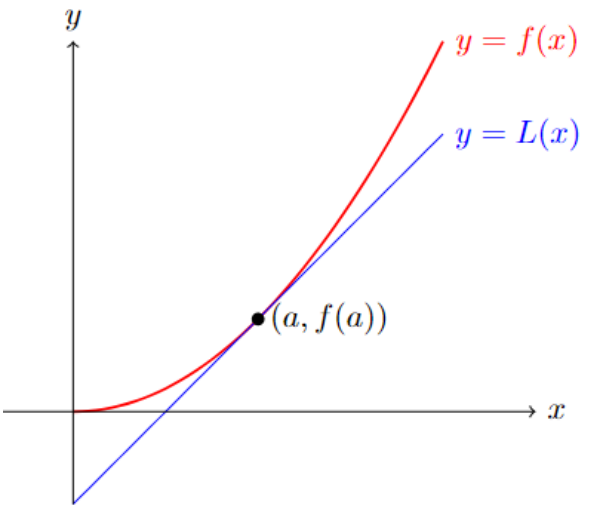
From the conclusion of MVT, we have $f(10) - f(2) = f'(c)(10 - 2)$, so $f(10) = f(2) + f'(c)(10 - 2) = -5 + 8f'(c)$.

Since $f'(x) \leq 6$ then $f'(c) \leq 6$. Thus $f(10) = -5 + 8f'(c) \leq -5 + 8(6) = 43$. Thus, the largest possible value for $f(10)$ is 43.

8.4.6 Linearization

Recall that $f'(a)$ is the slope of the tangent line of $f(x)$ at $x = a$. Conversely, given a function $f(x)$, the equation of the tangent line that touch $f(x)$ at $x = a$ using derivative $f'(a)$ is given by $L(x) = f(a) + f'(a)(x - a)$.

Look at the graph of a function $f(x)$ with its tangent line at $x = a$ below:



Note that the values of $f(x)$ and $L(x)$ get closer and closer as $x \to a$, so we can use $L(x)$ to approximate the value of $f(x)$ near $x = a$. Due to this, this tangent line $L(x)$ is often called the **linear approximation** or **linearization** of $f(x)$ at $x = a$.

Example

Find the linearization for $f(x) = \sqrt[3]{x}$ at $x = 27$, and use it to approximate $\sqrt[3]{27.137}$ and $\sqrt[3]{8}$.

Now $f(27) = \sqrt[3]{27} = 3$ and using power rule get $f'(x) = \frac{1}{3}x^{2/3} = \frac{1}{3\sqrt[3]{2}}$ so $f'(x) = \frac{1}{27}$. The linearization is then $L(x) = 3 + \frac{1}{27}(x - 27)$.

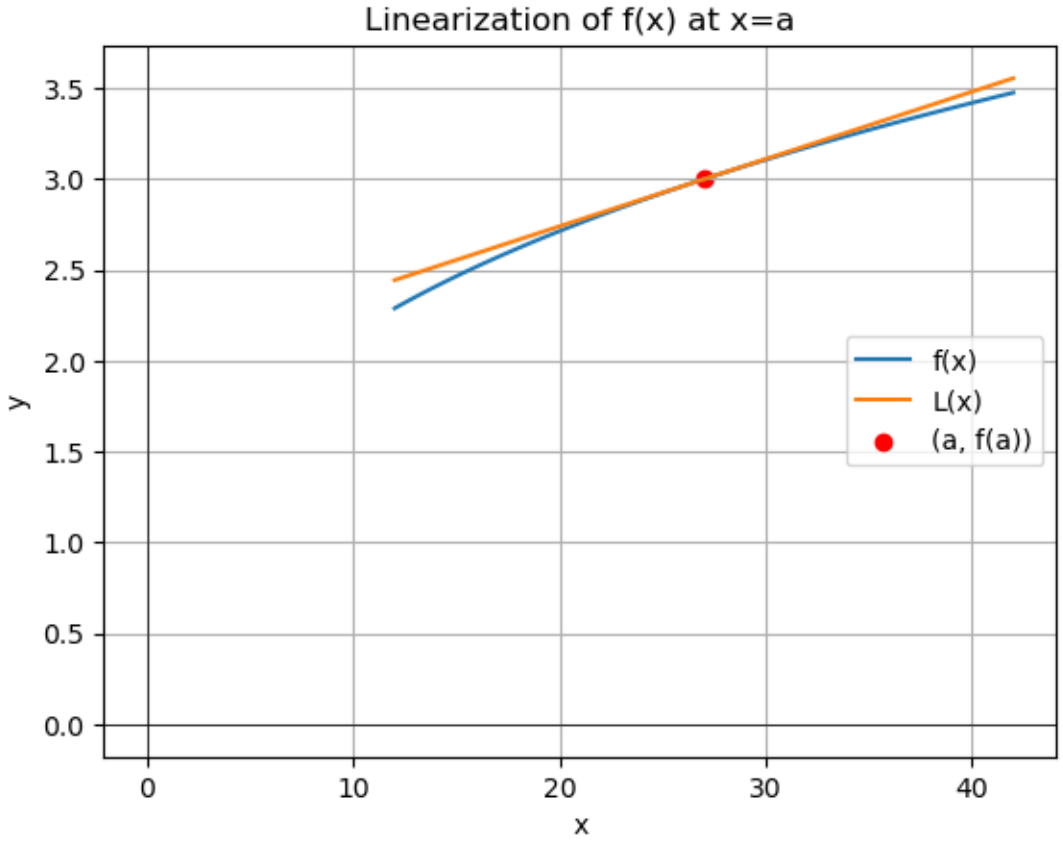
Thus $\sqrt[3]{27.137} \approx L(27.137) = 3 + \frac{1}{27}(0.137) = 3.00507\overline{407}$ and $\sqrt[3]{8} \approx L(8) = 3 + \frac{1}{27}(-19) = 2.\overline{296}$.

Comparing with the actual value $\sqrt[3]{27.137} \approx 3.005065516\ldots$ and $\sqrt[3]{8} = 2$, see that the approximation for $\sqrt[3]{27.137}$ is much more accurate than the approximation for $\sqrt[3]{8}$.

This makes sense since 8 is much further from 27 compared to 27.137 and hence the distance between the curve of $f(x)$ and the tangent line $L(x)$ is much further as well.

Below is the code that prompts the user to give the function $f(x)$ and the point $x = a$. The code return equation $L(x)$ at $x = a$ as well as their plots.

```
Equation of L(x): 0.037037037037037 x + 2.0
```



8.4.7 Implicit Differentiation

So far we have discussed derivatives of functions of the form $y = f(x)$. However, there are cases where the equations relating y and x either cannot be written explicitly as $y = f(x)$, or writing it as $y = f(x)$ results in some issues in finding the derivative. In these cases, we can use **implicit differentiation**.

Example

- Writing the equation $e^{x+2y} = \ln(xy^2) + x^2$ in the form $y = f(x)$ is challenging at best.
- Writing the equation $x^2 + y^2 = 4$ in the form $y = f(x)$ gives $y = \pm\sqrt{4 - x^2}$, but then how to deal with the $\pm$ for the derivative if we only want to write the derivative as a single function?

With implicit differentiation, we can treat the variable y as $y(x)$ and utilize Chain Rule as we derive both sides and obtain the derivative as a single function that likely will have both x and y. Below is the example for the equation $x^2 + y^2 = 4$:

$$\frac{d}{dx}\left(x^2 + [y(x)]^2\right) = \frac{d}{dx}[4] \tag{8.9}$$

$$2x + 2 \cdot y(x) \cdot y'(x) = 0$$

$$2x + 2yy' = 0$$

$$2yy' = -2x$$

$$y' = -\frac{x}{y}$$

While Python does not have built-in functionality specifically dedicated to implicit differentiation, we can leverage SymPy to perform implicit differentiation. Depending on the complexity of the equation, further algebraic manipulation or specific handling may be required. Below is a code example, prompting the user to input the equation. For $x^2 + y^2 = 4$, first rewrite into $x^2 + y^2 - 4 = 0$ and input the nonzero side as the equation:

```
The derivative dy/dx is: -x/y
```

8.4.8 Exercises

Exercises

1. (Section 4.1) Find the derivative of the following functions using the limit definition of the derivative.
   - $f(x) = 3x^2 - 12x + 32$
   - $g(x) = \dfrac{x}{x+2}$
   - $h(x) = \sqrt{4x^2 - 6}$
2. (Section 4.1) Prove $[f \pm g]' = f' \pm g'$ using the limit definition of the derivative.
3. (Section 4.2) Find the derivative of the following functions using the appropriate properties, formulas, and/or rules. Compare the result by using code.
   - $f(x) = 3x^2 - 12x + 32$
   - $g(x) = \dfrac{x}{x+2}$
   - $h(x) = \sqrt{4x^2 - 6}$
   - $z(x) = x^5 \sin\left(3x^4 + 1\right) + \ln\left(20^x - e^{5x}\right)$
4. (Section 4.2) Find the derivative of the following function using both Product and Quotient Rule, and verify that they gives the same answer.
   - $a(x) = \dfrac{\sin(x)}{x}$
   - $b(x) = \dfrac{\cos(x)}{2 - x^9}$
   - $c(x) = \dfrac{\sin\left(3 + 4^x\right)}{2 - x^9}$
5. (Section 4.2) Use Quotient Rule and the Basic Derivative formulas for $\sin(x)$ and $\cos(x)$, to find the derivative of $\tan(x)$, $\cot(x)$, $\sec(x)$, and $\csc(x)$.

6. (Section 4.3) Find the interval(s) in which the following functions is increasing, decreasing, concave up, concave down.
   - $a(x) = 2x^3 + 3x^2 - 12x + 4$ on $[-4, 3]$
   - $b(x) = 3x + \sin(4x)$ on $[0, 1]$
7. (Section 4.4) Find the absolute and relative extrema of the following functions:
   - $a(x) = 2x^3 + 3x^2 - 12x + 4$ on $[-4, 3]$
   - $b(x) = 3x + \sin(4x)$ on $[0, 1]$
8. (Section 4.5) Use the MVT to do the following problems:
   - Determine all the number(s) c which satisfy the conclusion of the MVT for the function $f(x) = 4x^3 - 8x^2 + 7x - 2$ on $[-3, 5]$
   - Given $f(x)$ a continuous and differentiable function on $[-3, 2]$. It is known that $f(-3) = 5$ and $f'(x) \geq 7$. What is the smallest possible value for $f(2)$?
9. (Section 4.6) Find the linearization for each given function at each given points, and plot the graph together with their linearization. For each, also use the linearization to approximate the value at the given nearby point, then calculate the error of the approximation.
   - $f(x) = \sqrt[4]{x}$ at $x = 16$, use to approximate $\sqrt[4]{16.7}$
   - $g(x) = \sin(x)$ at $x = \dfrac{\pi}{4}$, use to approximate $\sin\left(\dfrac{3\pi}{11}\right)$
10. (Section 4.7) Use implicit differentiation to find y' for each of the following:
    - $5x + x^4y^3 = 7y^3 - 2$
    - $3x^2 \tan(y) = x - y^5 \sec x$
    - $e^{x+2y} = \ln(xy^2) + x^2$

8.5 Integrals

Note

Below is the list of topics that are covered in this section:

- Antiderivative and Indefinite Integrals
- Area Problem and Definite Integrals
- Fundamental Theorems of Calculus
- Improper Integrals
- Average Value
- Area Between Curves

```
import sympy as sp
from sympy import ○○
```

8.5.1 Antiderivatives and Indefinite Integrals

In the previous section, we worked with derivatives. One easy example with power rule is that **the** derivative of $f(x) = x^2$ is $f'(x) = 2x$.

What if we would like to reverse this process? What is a function $f(x)$ that has derivative $2x$? Note that x^2 is not the only such function as $x^2 + 3, x^2 - 5, x^2 + 66, x^2 - \pi$, in general, $x^2 + C$ for any constant C, are all derived into $2x$.

Given a function $f(x)$, **an antiderivative** of $f(x)$ is any function $F(x)$ such that $F'(x) = f(x)$.

If $F(x)$ is an antiderivative of $f(x)$, then the general antiderivative of $f(x)$ is called an **indefinite integral**, denoted

$$\int f(x)\,dx = F(x) + C, \quad \text{where } C \text{ is a constant}$$

- The symbol $\int$ is called the **integral** symbol
- $f(x)$ is called the **integrand**
- x is called the **integration variable**
- C is called the **constant of integration**

In this case we say we are **integrating** $f(x)$ **with respect to (w.r.t.)** x.

Properties of the Indefinite Integral

- Constant multiples can be factored out the indefinite integral

$$\int k \cdot f(x)\,dx = k \cdot \int f(x)\,dx$$

- Indefinite integral of the sum or difference of functions can be broken up

$$\int f(x) \pm g(x)\,dx = \int f(x)\,dx \pm \int g(x)\,dx$$

Basic Antiderivative Formulas

Antiderivative formulas for some basic integrands:

- Constant

$$\int k\,dx = kx + C$$

- Power Rule

$$\int x^n\,dx = \frac{x^{n+1}}{n+1} + C, \quad n \neq -1$$

- Power of -1

$$\int \frac{1}{x}\,dx = \ln|x| + C$$

- Exponential

$$\int b^x\,dx = \frac{b^x}{\ln(b)} + C$$

- Exponential when $b = e$

$$\int e^x \, dx = \frac{e^x}{\ln(e)} + C = \frac{e^x}{1} + C = e^x + C$$

- Sine

$$\int \sin(x) \, dx = -\cos(x) + C$$

- Cosine

$$\int \cos(x) \, dx = \sin(x) + C$$

Integration by U-substitution

In the Derivative section, we saw how Chain Rule is used to find the derivative of a composition of two functions. Similarly, there is a method that can be used to find an antiderivative of certain functions involving the composition of functions.

Rule:

$$\int f\left(g\left(x\right)\right) \cdot g'(x) \, dx = \int f(u) \, du, \quad \text{where } u = g(x)$$

Note that to use this rule, the integrand must contain a composition function $f\left(g\left(x\right)\right)$ multiplied with the derivative of the inside function $g(x)$.

Steps for U-subs

1. Find u, which is the inside function
2. Find du in terms of x and dx
3. Substitute the integrals, if done correctly will only have integration in terms of variable u, not the original variable.
4. Integrate
5. Substitute back the u in terms of the original variable.

Example

Determine $\int 6x^2 \left(x^3 + 8\right)^5 \, dx$.

The "inside function" here is $x^3 + 8$, whose derivative is $3x^2$. Hence, set up the integration as such, then do the steps above:

$$\int 6x^2 \left(x^3 + 8\right)^5 \, dx = 2 \cdot \int \left(x^3 + 8\right)^5 \cdot 3x^2 \, dx$$

Steps:

1. $u = x^3 + 8$
2. $du = 3x^2 \, dx$
3. Substitute in u: $2 \cdot \int \underbrace{\left(x^3 + 8\right)}_{u}^{5} \cdot \underbrace{3x^2 \, dx}_{du} = 2 \cdot \int u^5 \, du$
4. Integrate: $2 \cdot \int u^5 \, du = 2 \cdot \frac{u^6}{6} + C$
5. Substitute back to x: $2 \cdot \frac{u^6}{6} + C = 2 \cdot \frac{\left(x^3 + 8\right)^6}{6} + C = \frac{\left(x^3 + 8\right)^6}{3} + C$

Hence:

$$\int 6x^2 \left(x^3 + 8\right)^5 \, dx = \frac{\left(x^3 + 8\right)^6}{3} + C$$

Integration by Parts (IBP)

In the Derivative section, we saw how Product Rule is used to find the derivative of a product of two functions. Similarly, there is a method that can be used to find an antiderivative of certain functions involving the product of two functions.

Rule

$$\int f \cdot g' \, dx = fg - \int f' \cdot g \, dx$$

or, more commonly remembered as:

$$\int u \, dv = uv - \int v \, du$$

A common issue with this setup is deciding which factor of the product should be u and which one should be dv. While there are many "tricks" offered by some resources, there are no official rules on which factor is which. If one setup does not work, we can always try the other setup. Repetitive practice is the best way to develop the sense of recognizing which factor is "a good choice" to be u or dv.

Example

Determine $\int x \cdot e^{3x} \, dx$

First choose the u and dv:

$$u = x \qquad dv = e^{3x} \, dx$$

Next, find du and v:

$$u = x \Rightarrow du = dx$$
$$dv = e^{3x} \Rightarrow \int e^{3x} \, dx = \frac{e^{3x}}{3}$$

Thus:

$$\begin{aligned} \int x \cdot e^{3x} \, dx &= x \cdot \frac{e^{3x}}{3} - \int \frac{e^{3x}}{3} \, dx \\ &= \frac{xe^{3x}}{3} - \frac{1}{3} \cdot \int e^{3x} \, dx \\ &= \frac{xe^{3x}}{3} - \frac{1}{9} e^{3x} + C \end{aligned}$$

Now, try to do $u = e^{3x}$ and $dv = x \, dx$ instead. What happens?

Note that both U-Subs and IBP are used to work on the product of two functions, with the restriction on U-Subs being stricter. Hence, there are indefinite integrals that can be evaluated using either U-Subs and IBP.

Below is a code example that asks the user for the integrand $f(x)$, and returned the general antiderivative $\int f(x) \, dx$.

```
def find_antiderivative():
    # Prompt the user for the function f(x)
    function_input = input("Enter the function f(x): ")

    # Define the variable and function using sympy
    x = sp.Symbol('x')
    f = sp.sympify(function_input)

    # Compute the antiderivative of f(x)
```

(continues on next page)

(continued from previous page)

```
    antiderivative = sp.integrate(f, x)

    # Return the general antiderivative
    return antiderivative

# Run the function to find the antiderivative
antiderivative = find_antiderivative()

# Print the general antiderivative
print("General Antiderivative:")
print(antiderivative)
```

```
General Antiderivative:
-cos(2*x + 7)/2
```

8.5.2 Area Problem and Definite Integrals

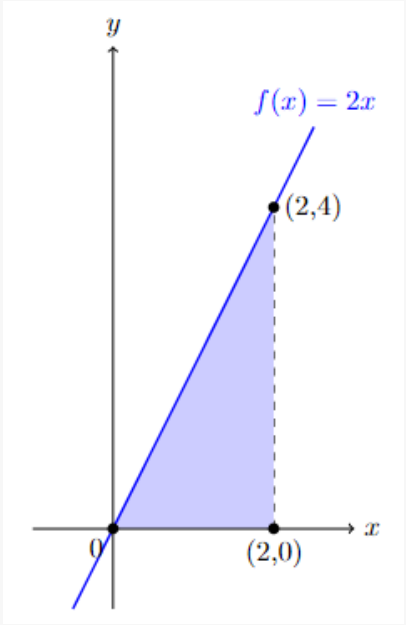
Unlike indefinite integrals, definite integrals don't come from reversing the process of evaluating derivatives. Definite integrals come from **Area Problems**. Analogously, the area problems to definite integrals are like RoC problems to derivatives.

The goal of the area problem is to find the area of the region between the graph of $f(x)$ and the x-axis. If that region has a common two-dimensional shape, then it is enough to use basic geometry to find the area. If it doesn't, then it can be solved analytically using definite integrals.

Example

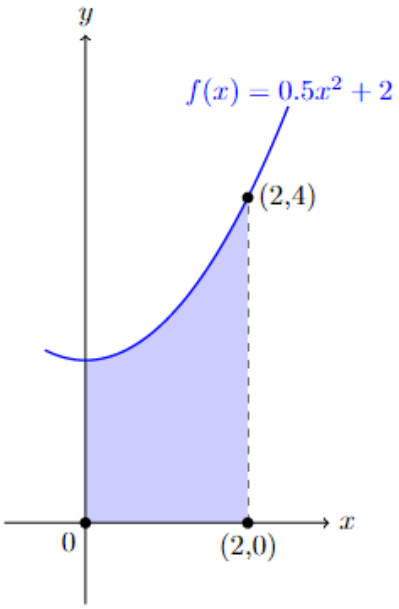
Find the shaded area formed by the function $f(x) = 2x$ and the x-axis on the interval $[0, 2]$.

Note that the area is represented with a simple triangle with base 2 and height 4. Hence the area is $\frac{1}{2} \cdot 2 \cdot 4 = 4$.



Now, how about finding the shaded area formed by the function, say, $f(x) = 0.5x^2 + 2$ and the x-axis on the interval

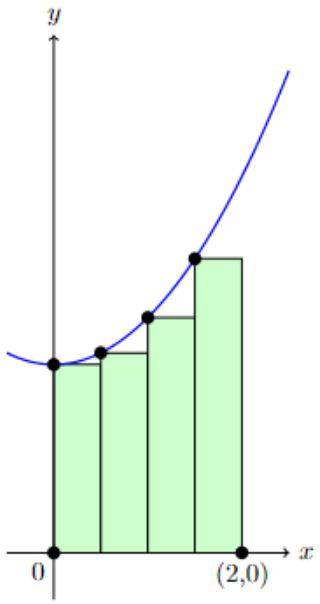
$[0, 2]$?



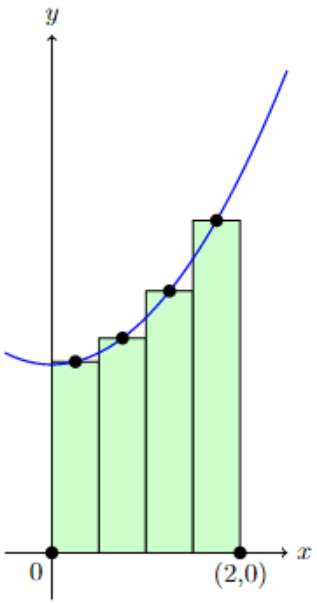
While we might not able to calculate this area exactly using high school 2D geometry, we can do an estimation.

Estimating the area can be done by dividing the interval into several subintervals, and in each subinterval create a rectangle whose height is using the function value at a specific point in that subinterval. The approximation of the area is then obtained by adding the total area of all rectangles.

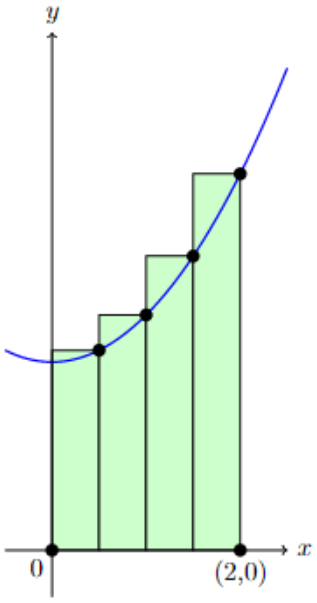
Below is the example if we divide the interval $[0, 2]$ into 4 subintervals, and use certain points as the height of each rectangle:



Left Sum



Middle Sum



Right Sum

As their name suggests, the **Left Sum, Middle Sum (Midpoint Rule)**, and **Right Sum** are using the function values at the left endpoints, midpoints, and right endpoints, respectively, as the height of the rectangles.

Since this example divides the interval $[0, 2]$ into 4 subintervals, then the width of each rectangle is $\frac{2-0}{4} = 0.5$.

Let A_L, A_M,and A_R denote the area approximation using the Left, Middle, and Right sum, respectively. Then:

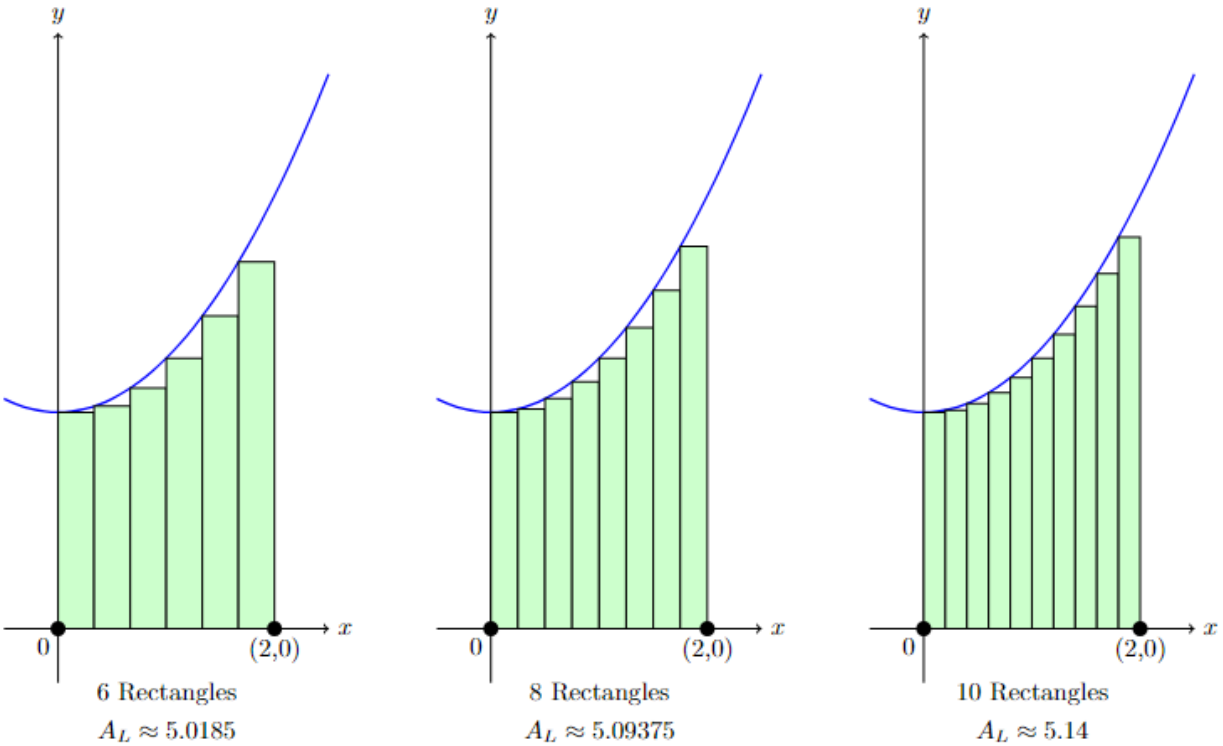
$$\begin{aligned} A_L &= 0.5 \cdot f(0) + 0.5 \cdot f(0.5) + 0.5 \cdot f(1) + 0.5 \cdot f(1.5) \\ &= 0.5 \cdot [f(0) + f(0.5) + f(1) + f(1.5)] \\ &= 0.5 \cdot [2 + 2.125 + 2.5 + 3.125] = 4.875 \end{aligned}$$

$$\begin{aligned} A_M &= 0.5 \cdot f(0.25) + 0.5 \cdot f(0.75) + 0.5 \cdot f(1.25) + 0.5 \cdot f(1.75) \\ &= 0.5 \cdot [f(0.25) + f(0.75) + f(1.25) + f(1.25)] \\ &= 0.5 \cdot [2.03125 + 2.28125 + 2.78125 + 3.53125] = 5.3125 \end{aligned}$$

$$\begin{aligned} A_R &= 0.5 \cdot f(0.5) + 0.5 \cdot f(1) + 0.5 \cdot f(1.5) + 0.5 \cdot f(2) \\ &= 0.5 \cdot [f(0.5) + f(1) + f(1.5) + f(2)] \\ &= 0.5 \cdot [2.125 + 2.5 + 3.125 + 4] = 5.875 \end{aligned}$$

In this particular example, the Left Sum clearly gives an underestimate as all the rectangles' heights are below the graph of $f(x)$. Meanwhile, the Right Sum gives an overestimate as all the rectangles' heights are above it. The Midpoint Rule seems to give the least error from the figure as the height of the rectangles are closer to the value on the actual graph.

Illustrated below is the approximation using Left Sum on 6, 8, and 10 rectangles:



Even though graphically we can see the increase in accuracy when the number of rectangles increases, how should we know the actual area?

Let n denote the number of rectangles, Δx denote the width of each rectangle, and $x_1, x_2, \ldots, x_n$ denote the points in the interval $[a, b]$ whose values on those points are used as the height of the rectangle.

The area approximation is then:

$$A \approx \Big[f(x_1) + f(x_2) + \ldots + f(x_n) \Big] \cdot \Delta x \qquad \text{or} \qquad A \approx \sum_{i=1}^{n} f(x_i) \cdot \Delta x$$

The summation above is what we call **Riemann Sum**.

Increasing the number of rectangles n gives a better approximation of the area. In fact, taking the limit as $n \to \infty$ gives the actual area.

This limit of Riemann Sum as n goes to infinity is the **definite integral**, denoted as follows:

$$\int_a^b f(x)\,dx = \lim_{n\to\infty} \sum_{i=1}^{n} f(x_i) \cdot \Delta x$$

For our previous example $f(x) = 0.5x^2 + 2$ and the x-axis on the interval $[0, 2]$:

For n subintervals, the width of each subinterval is $\Delta x = \dfrac{2-0}{n} = \dfrac{2}{n}$, and hence the subintervals are:

$$\left[0, \frac{2}{n}\right], \left[\frac{2}{n}, \frac{4}{n}\right], \left[\frac{4}{n}, \frac{6}{n}\right], \ldots, \left[\frac{2(i-1)}{n}, \frac{2i}{n}\right], \ldots, \left[\frac{2(n-1)}{n}, 2\right]$$

As n is large, without loss of generality we can take the right endpoint as the height of each rectangle. The right endpoint of the ith subinterval is $x_i = \dfrac{2i}{n}$.

The Riemann Sum then becomes:

$$\begin{aligned}
\sum_{i=1}^{n} f(x_i) \cdot \Delta x &= \sum_{i=1}^{n} f\left(\frac{2i}{n}\right)\left(\frac{2}{n}\right) \\
&= \sum_{i=1}^{n} \left(0.5\left(\frac{2i}{n}\right)^2 + 2\right)\left(\frac{2}{n}\right) \\
&= \sum_{i=1}^{n} \left(\frac{4i^2}{n^3} + \frac{4}{n}\right) \\
&= \frac{4}{n^3}\sum_{i=1}^{n}(i^2) + \frac{4}{n}\sum_{i=1}^{n}(1) \\
&= \frac{4}{n^3}\left(\frac{n(n+1)(2n+1)}{6}\right) + \frac{4}{n}(n) \\
&= \frac{16n^2 + 6n + 2}{3n^2} \\
&= \frac{16}{3} + \frac{2}{n} + \frac{2}{3n^2}
\end{aligned}$$

Computing the integral:

$$\begin{aligned}
\int_0^2 0.5x^2 + 2\,dx &= \lim_{n\to\infty} \sum_{i=1}^{n} f(x_i) \cdot \Delta x \\
&= \lim_{n\to\infty} \left(\frac{16}{3} + \frac{2}{n} + \frac{2}{3n^2}\right) \\
&= \frac{16}{3} + 0 + 0 = \frac{16}{3}
\end{aligned}$$

The work can be tedious even for a simple function like $f(x) = 0.5x^2 + 2$. In the next part, we will learn a simpler way to do this area calculation.

Below is the code to calculate Riemann Sum of $f(x)$ on the interval $[a, b]$ using n intervals.

```
The Riemann sum is: 5.140000000000001
```

Properties of Definite Integral

Before going to the next part, familiarize yourself with the properties of the definite integral below, and think where they came from based on the relation with the area under the curve.

- $\int_a^b f(x)\,dx = -\int_b^a f(x)\,dx$
- $\int_a^a f(x)\,dx = 0$
- $\int_a^b cf(x)\,dx = c\int_a^b f(x)\,dx$ for any constant c
- $\int_a^b f(x) \pm g(x)\,dx = \int_a^b f(x)\,dx \pm \int_a^b g(x)\,dx$
- $\int_a^b f(x)\,dx = \int_a^c f(x)\,dx + \int_c^b f(x)\,dx$ for any constant c
- If $f(x) \geq g(x)$ for $a \leq x \leq b$ then $\int_a^b f(x)\,dx \geq \int_a^b g(x)\,dx$
- $\left|\int_a^b f(x)\,dx\right| \leq \int_a^b |f(x)|\,dx$

8.5.3 Fundamental Theorems of Calculus (FTC)

FTC Part 1

Theorem (FTC 1)

For $f(t)$ continuous function and a any constant: $\dfrac{d}{dx}\int_a^x f(t)\,dt = f(x)$.

While many 'resources' said that FTC 1 "basically" shows that derivative and integral "cancel each other", it is not actually that simple.

Motivated students are encouraged to explore the proof of FTC 1 (Hint: Use the Extreme Value Theorem).

Example

If $F(x) = \int_{-2}^x e^{3t^2}\sin^2(2-t^5)$, find $F'(x)$.

By FTC 1, $F'(x) = \dfrac{d}{dx}F(x)\,dx = \dfrac{d}{dx}\int_{-2}^x e^{3t^2}\sin^2(2-t^5)\;dx = e^{3x^2}\sin^2(2-x^5)$.

When the lower bound is not a constant, or if the upper bound is a more complicated function of x, FTC 1 can still be applied with the help of Chain Rule and integral properties.

- $\dfrac{d}{dx}\displaystyle\int_a^{u(x)} f(t)\,dt = u'(x)f(u(x))$
- $\dfrac{d}{dx}\displaystyle\int_{v(x)}^{u(x)} f(t)\,dt = -v'(x)f(v(x)) + u'(x)f(u(x))$

Example

$$\frac{d}{dx}\int_5^{x^2} \frac{t^3-7}{t^2+8}\,dt = 2x\frac{(x^2)^3-7}{(x^2)^2+8}.$$

FTC Part 2

FTC 2 connect area problem (definite integral) with antiderivative (indefinite integral).

It provides a simpler way to calculate definite integral compared to using the limit of Riemann Sum.

Theorem (FTC 2)

If $F(x)$ is an antiderivative of $f(x)$ and $f(x)$ is continuous on $[a,b]$, then: $\displaystyle\int_a^b f(x)\,dx = F(x)\Big|_a^b = F(b)-F(a)$.

Example

Let's visit the example used previously when we discuss Riemann Sum: Calculate $\displaystyle\int_0^2 0.5x^2+2\,dx$

Using basic antiderivative formula (in this case, for constant and power function), we have $\displaystyle\int 0.5x^2+2\,dx = \frac{0.5x^3}{3}+2x$, hence by FTC 2:

$$\int_0^2 0.5x^2+2\,dx = \left(\frac{0.5x^3}{3}+2x\right)\Big|_0^2 = \left(\frac{0.5(2)^3}{3}+2(2)\right)-\left(\frac{0.5(0)^3}{3}+2(0)\right) = \frac{4}{3}+4 = \frac{16}{3}$$

This gives the same answer with much less effort compared to using the limit of a Riemann Sum.

Below is an example code that prompts the user for $f(x)$ and $[a,b]$, then return the value of $\displaystyle\int_a^b f(x)\,dx$.

```
The definite integral is: 5.33333333333333
```

8.5.4 Improper Integrals

Consider the area problem on an unbounded interval $(a, \infty]$ or $(-\infty, b]$ or $(-\infty, \infty)$ instead of $[a, b]$. The definite integral that corresponds to the area problem over such unbounded interval is called **improper integral**.

To calculate improper integral, we first bound the interval to some number N, then take $N \to \pm\infty$ as needed:

- $\displaystyle\int_a^\infty f(x)\,dx = \lim_{N\to\infty}\int_a^N f(x)\,dx$
- $\displaystyle\int_{-\infty}^b f(x)\,dx = \lim_{N\to-\infty}\int_N^b f(x)\,dx$
- $\displaystyle\int_{-\infty}^\infty f(x)\,dx = \int_{-\infty}^c f(x)\,dx + \int_c^\infty f(x)\,dx = \lim_{N\to-\infty}\int_N^c f(x)\,dx + \lim_{N\to\infty}\int_c^N f(x)\,dx$, for any number c.

Since the area is over an unbounded interval, the area is potentially, but not necessarily, unbounded.

If the limit exists, then we say the improper integral is **convergent**. Otherwise, we say the improper integral is **divergent**.

Example

Determine if the following integrals are convergent or divergent: $\displaystyle\int_{-\infty}^\infty xe^{-x^2}\,dx$.

Using u-subs with $u = -x^2$, we have $\displaystyle\int xe^{-x^2}\,dx = -\frac{1}{2}e^{-x^2} + C$. Hence for the improper integral:

$$\begin{aligned}
\int_{-\infty}^\infty xe^{-x^2}\,dx &= \int_{-\infty}^0 xe^{-x^2}\,dx + \int_0^\infty xe^{-x^2}\,dx \\
&= \lim_{N\to-\infty}\int_N^0 xe^{-x^2}\,dx + \lim_{N\to\infty}\int_0^N xe^{-x^2}\,dx \\
&= \lim_{N\to-\infty}\left(-\frac{1}{2}e^{-x^2}\right)\Big|_N^0 + \lim_{N\to\infty}\left(-\frac{1}{2}e^{-x^2}\right)\Big|_0^N \\
&= \lim_{N\to-\infty}\left[\left(-\frac{1}{2}e^0\right) - \left(-\frac{1}{2}e^{-N^2}\right)\right] + \lim_{N\to\infty}\left[\left(-\frac{1}{2}e^{-N^2}\right) - \left(-\frac{1}{2}e^0\right)\right] \\
&= -\frac{1}{2} + \frac{1}{2} = 0
\end{aligned}$$

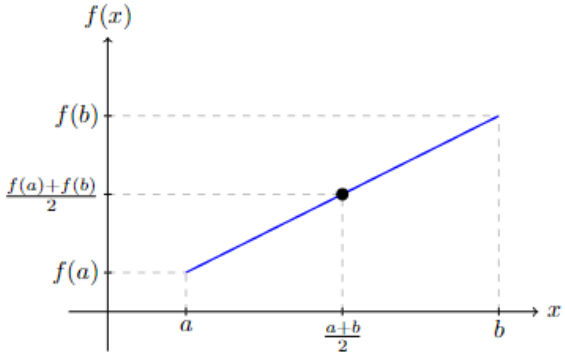
A little modification on sympy can be used to enable the code to calculate improper integrals. Below is the modified code so that the user can put either **-inf** (for $-\infty$) or **inf** (for ∞) as the integral bounds:

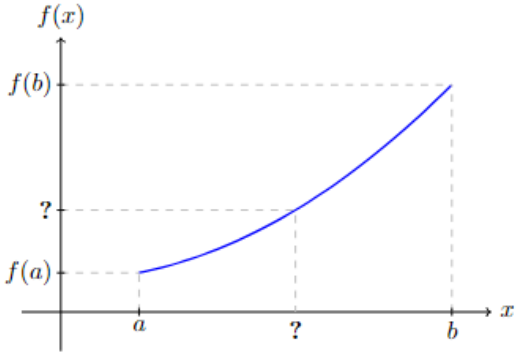
```
The definite integral is: 0
```

8.5.5 Average Value

When a continuous function $f(x)$ is linear, the average value $f(x)$ on $[a, b]$ is simply $\frac{f(a)+f(b)}{2}$.

How about the average value of nonlinear continuous function $f(x)$?

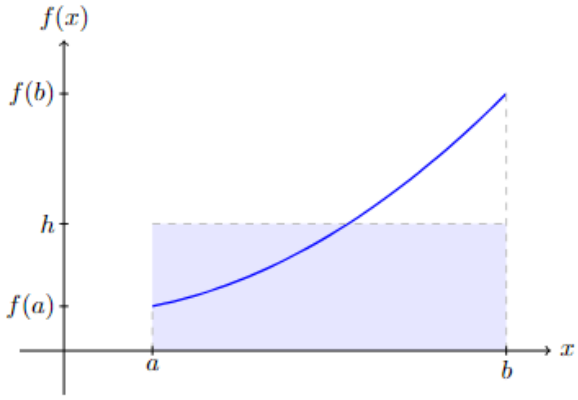




The **average value** of continuous function $f(x)$ on $[a, b]$ is given by $\dfrac{1}{b-a}\displaystyle\int_a^b f(x)\,dx$. Motivated students are encouraged to try proving this formula algebraically using Riemann Sum.

An easier way to look at the average value is the height h such that the area of the rectangle with width $(b-a)$ and height h is equal to the area between $f(x)$ and the x-axis on $[a, b]$:

$$h(b-a) = \int_a^b f(x)\,dx \quad \Rightarrow \quad h = \frac{1}{b-a}\int_a^b f(x)\,dx$$



Example

Determine the average value of $f(x) = 5\cos(3x) + x$ on $[1, 4]$.

Average value is

$$\begin{aligned}\frac{1}{4-1}\int_1^4 5\cos(3x) + x\,dx &= \frac{1}{3}\left(\frac{5}{3}\sin(3x) + \frac{x^2}{2}\right)\Bigg|_1^4 \\ &= \frac{1}{3}\left[\left(\frac{5}{3}\sin(12) + \frac{16}{2}\right) - \left(\frac{5}{3}\sin(3) + \frac{1}{2}\right)\right] \approx 2.12.\end{aligned}$$

If want to ask the user for $f(x)$ and find the average value for $f(x)$ on $[a, b]$, remember to check the continuity of $f(x)$ on $[a, b]$.

```
import sympy as sp

def calculate_average_value():
    x = sp.Symbol('x')

    # Prompt user for function f(x) and the interval [a,b]
    f_x_str = input("Enter the function f(x): ")
    f_x = eval(f_x_str)  # Evaluate the function string as a valid Python expression
    a = float(input("Enter the left endpoint of the interval (a): "))
    b = float(input("Enter the right endpoint of the interval (b): "))

    # Check if f(x) is not continuous on [a, b]
    is_continuous = True
    for point in [a, b]:
        if not sp.limit(f_x, x, point, dir='-').equals(sp.limit(f_x, x, point, dir='+
↪')):
            is_continuous = False
            break

    if not is_continuous:
        return "f(x) is not continuous on [a,b]"

    # Calculate the average value of f(x) on [a,b]
    avg_value = sp.integrate(f_x, (x, a, b)) / (b - a)
    return avg_value

# Test the function
result = calculate_average_value()
print(result)
```

```
2.12350392996650
```

The Mean Value Theorem for Integrals

Theorem (MVT for Integrals): If $f(x)$ is continuous on $[a, b]$, then there is a number c in $[a, b]$ such that $\int_a^b f(x)\,dx = f(c)\,(b-a)$.

Example

Find the number c that satisfies the MVT for Integrals, for $f(x) = 3x^2 + 5x - 7$ on $[2, 4]$.

Since $f(x) = 3x^2 + 5x - 7$ is a polynomial and hence is continuous on $[2, 4]$, apply the MVT for Integrals:

$$\int_2^4 3x^2 + 5x - 7\,dx = (3c^2 + 5c - 7)\,(4 - 2)$$

$$\left(x^3 + \frac{5x^2}{2} - 7x\right)\Bigg|_2^4 = 2\,(3c^2 + 5c - 7)$$

$$72 = 2\,(3c^2 + 5c - 7)$$

$$36 = 3c^2 + 5c - 7$$

$$0 = 3c^2 + 5c - 43$$

Using quadratic formula we get $c = \dfrac{-5 \pm \sqrt{541}}{6}$, or approximately, $c \approx -4.7099, 3.0432$.

The only c value that is in the given interval $[2, 4]$ is $c \approx 3.0432$.

Below is the example code that ask user for a continuous $f(x)$ on $[a, b]$, and will return the value(s) of c that satisfy the MVT for integrals.

```
The value(s) of c that satisfy MVT for Integrals on [a,b]:
3.04323444987100
```

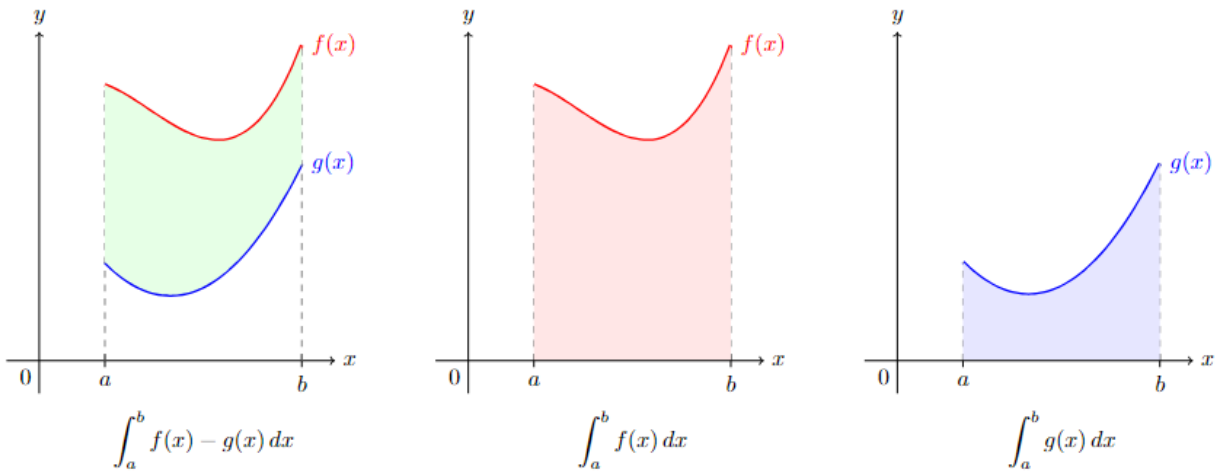
8.5.6 Area Between Curves

We already know that the definite integral $\int_a^b f(x)\,dx$ gives the area between the graph of $f(x)$ and the x-axis over interval $[a, b]$.

Now we want to calculate the area between the graph of two functions.

In general, the area between two functions over the interval $[a, b]$ is $A = \int_a^b (\text{top function}) - (\text{bottom function})\,dx$.

To illustrate the idea, consider a specific case where two continuous functions $f(x)$ and $g(x)$ satisfies $f(x) \geq g(x)$ on $[a, b]$ with the diagram below.



When the functions $f(x)$ and $g(x)$ alternate the upper and bottom position, one common way to find the area is to separate the interval $a \leq x \leq b$ into multiple subintervals where in each subinterval the top and bottom position is consistent. However, the shorter way to do it is to keep the integrand positive using absolute value:

$$A = \int_a^b \Big| f(x) - g(x) \Big| \, dx$$

This formula holds no matter which function $f(x)$ of $g(x)$ is on the top or bottom.

Similarly, the area between $x = f(y)$ and $x = g(y)$ on the interval $c \leq y \leq d$ is $A = \int_c^d (\text{right function}) - (\text{left function}) \, dx$.

In general, the formula is $A = \int_c^d \Big| f(y) - g(y) \Big| \, dy$.

Below is example code that asks the user for $f(x)$, $g(x)$, and $[a, b]$. It will return the area between the two $f(x)$ and $g(x)$ over the interval $[a, b]$.

```
The area between f(x) and g(x) over the interval [a, b] is:
3.509157819444367
```

8.5.7 Exercises

Exercises

1. (Section 5.1) Find the indefinite integral of the following functions.
   - $f(x) = \sin(\cos(x))\sin(x)$
   - $g(x) = 3x\sqrt[3]{1-2x^2}$
   - $h(x) = \dfrac{3x^5 + 2x}{\left(x^6 + 2x^2\right)^3}$
   - $m(x) = x^5\sqrt{x^3+1}$
     - Hint: if doing it manually, use u-subs twice in the process
   - $n(x) = x^2\cos(7x)$
     - Hint: if doing it manually, use IBP twice in the process
2. (Section 5.3) Use FTC 2 to compute the following, and determine which Sum (Left/Midpoint/Right) on the previous problem gives the best approximation.
   - $\int_0^{\pi} \sin(x)\,dx$
   - $\int_{-2}^{6} 3x^3 + 5x - 7\,dx$
   - $\int_2^{10} x^5\sqrt{x^3+1}\,dx$
3. (Section 5.3) Use FTC 1 to differentiate each of the following integrals with respect to x.
   - $\int_2^x 7\sin^2\left(t^3 - 0.2t + 9\right)\,dt$

- $\int_{2}^{\cos(3x)} \sqrt{17 - 9t^6}\, dt$
- $\int_{-5x}^{5x^2} e^t - \ln(4t)\, dt$

4. (Section 5.4) Determine if the following integral is convergent or divergent. If convergent, calculate its value. If divergent, explain why.
   - $\int_{4}^{\infty} \frac{2}{x^3}\, dx$
   - $\int_{-\infty}^{5} e^{2x} + 1\, dx$
   - $\int_{-\infty}^{\infty} \frac{x^2}{x^3 + 1}\, dx$
5. (Section 5.5) Find the average value of the given function on the given interval.
   - $f(x) = 10\ln(x) + x^2$ on $[3, 5]$
   - $t(x) = 70 + 950e^{-0.07x}$ on $[0, 60]$
   - $g(x) = -\sin\left(\frac{1}{2}x\right) + \cos(2x)$ on $\left[\frac{\pi}{2}, \pi\right]$
6. (Section 5.5) Find the number c that satisfies the MVT for Integrals, for $f(x) = 4x^2 + 5x + 6$ on $[-1, 5]$.
7. (Section 5.6) Find the area of the following:
   - between $f(x) = 3x$ and $g(x) = x\,(\sqrt{x} + 1)$ for $0 \leq x \leq 4$
   - between $f(y) = 3y$ and $g(y) = 30y \cdot e^{-0.3y}$ for $0 \leq y \leq 9$
   - enclosed by $f(x) = 4 - x$ and $g(x) = (x - 1)^2$
     - Hint: find where $f(x)$ and $g(x)$ intersects

8.6 Parametric Equations

Tip

Below is the list of topics that are covered in this section:

- Parametric Equations - Curves and Tangents
- Parametric Equations - Area and Arc Length
- Polar Coordinates - Conversion and Tangents
- Polar Coordinates - Area and Arc Length

```
import numpy as np
import sympy as sp
import matplotlib.pyplot as plt
from scipy.integrate import quad
import math
```

8.6.1 Parametric Equations - Curves and Tangents

As we already discussed in Implicit Differentiation in the Derivative Section, many functions cannot be written exclusively as $y = f(x)$ or $x = h(y)$.

A simple example is the equation of a circle that is centered at the origin, $x^2 + y^2 = r^2$. Trying to write this equation exclusively gives $y = \pm\sqrt{r^2 - x^2}$ or $x = \pm\sqrt{r^2 - y^2}$, but these contain two functions each.

Besides circles, there are many common curves out there that cannot be written as a single equation in terms of x or y (can you think of some other examples?)

To make it easier to work with such curves, **parametric equations** are introduced. The idea is to involve a third variable, called a **parameter**, usually (but not necessarily) denoted with t, such that $x = f(t)$ and $y = g(t)$.

Parametric Curves

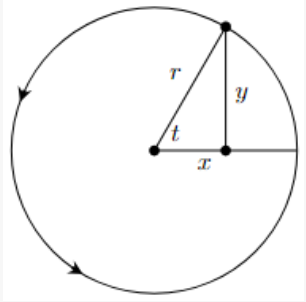
Parametric Curve is the collection of points $(x, y) = (f(t), g(t))$ obtained by letting t be *all possible values*.

When *all possible values* of t is not restricted, there is really no official *rule* on which values of t need to be chosen. Some simply try many different values of t until they are *confident enough* that they have enough information to know what the shape of the curve should be. Moreover, there are cases where many values of t give the same point on the curve so the restriction can be quite natural. To describe this, let's use the circle equation as an example.

Example

Show that the parametric equations $x = r\cos(t)$ and $y = r\sin(t)$ are equivalent to the circle equation $x^2 + y^2 = r^2$. Can t be restricted?

To prove this, consider the unit following circle:



In the right triangle, we can see that $\sin(t) = \dfrac{y}{r}$ and $\cos(t) = \dfrac{x}{r}$.

Combine with the trigonometric identity $\sin^2(t) + \cos^2(t) = 1 \quad \Rightarrow \quad \left(\dfrac{y}{r}\right)^2 + \left(\dfrac{x}{r}\right)^2 = 1.$

Removing the parentheses we get $\dfrac{y^2}{r^2} + \dfrac{x^2}{r^2} = 1$ and then multiplying both sides by r^2 we get $x^2 + y^2 = r^2$.

The arrows on the parametric curves indicate which way the curve goes as t increases, and are called the **direction of motion**. While they are helpful to see the direction, the arrows are not always shown as they are not an actual part of the curve itself.

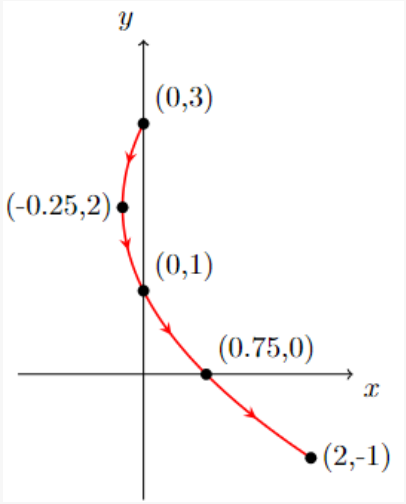
Here t can be restricted to $0 \le t \le 2\pi$, as taking value $t < 0$ or $t > 2\pi$ just overwrites parts of the same circle curve. Of course, this kind of restriction doesn't occur in general.

Example

Sketch the parametric curve for the parametric equations $x = t + t^2$ and $y = 1 - 2t$ on $-1 \leq t \leq 1$.

We can start by picking certain values of t on $[-1, 1]$ and get some points (t, x, y) : $(-1, 0, 3)$, $(-0.5, -0.25, 2)$, $(0, 0, 1)$, $(0.5, 0.75, 0)$, $(1, 2, -1)$.

Plotting the points with the direction of motion, and connecting the points give the parabola:



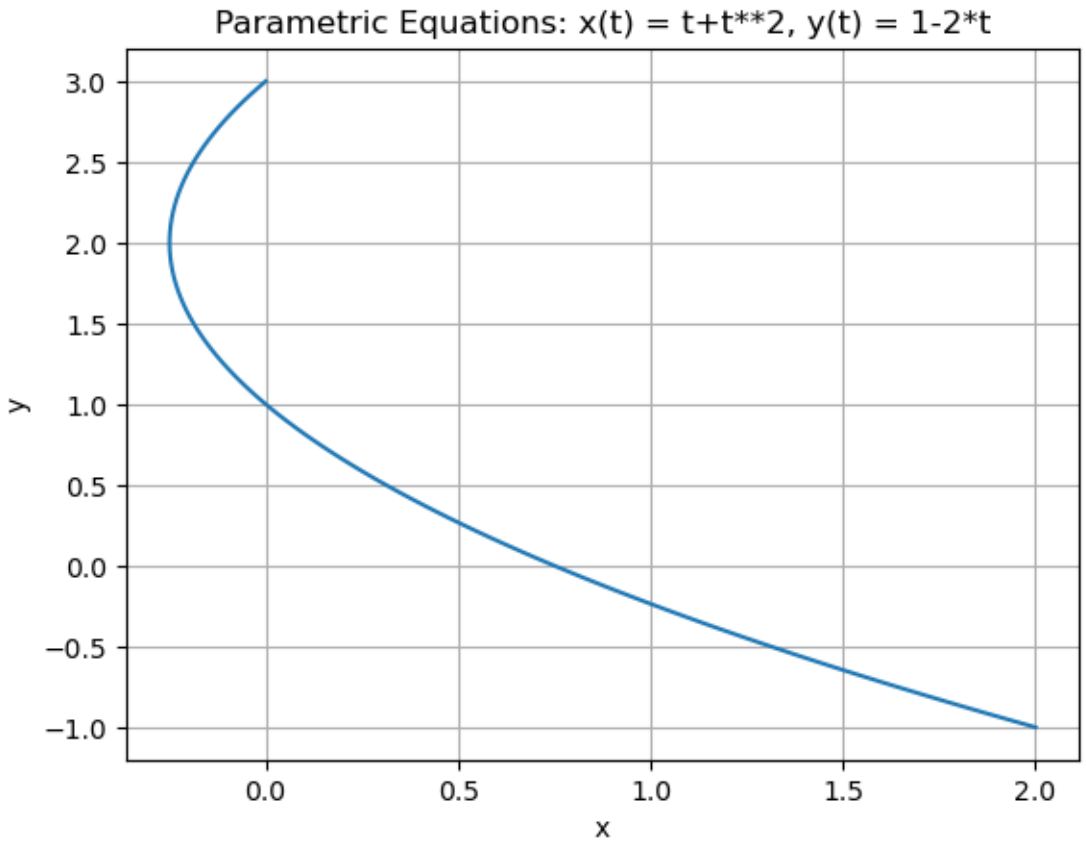
However, how does one know that connecting the points gave a parabola while we never check where the points are for the other t values in $[-1, 1]$?

Particularly in this case, we can confirm that the curve is a parabola by eliminating the parameter t in the equations:

$$y = 1 - 2t \Rightarrow t = 1 - \frac{y}{2}$$
$$x = t + t^2 \Rightarrow x = \left(1 - \frac{y}{2}\right) + \left(1 - \frac{y}{2}\right)^2$$

From here we can see that x is a quadratic equation of y, confirming the shape of the curve.

However, the parameter cannot always be easily eliminated from the parametric equations like in the above example. Even when it can be eliminated, it does not always give an easily recognizable equation that can confirm the curve like in the above example. That is why there are cases where the problem solver must pick *enough* values of t until they are *confident enough* to know what the shape of the curve is. Using a graphing tool, such as Python, makes this simpler. Below is the code to plot the curve of parametric equations $x(t)$ and $y(t)$ on $a \leq t \leq b$.



Tangents with Parametric Equations

Recall in the Derivative section that for $y = f(x)$, we can find the tangent line at $x = a$ by using linearization $y = L(x) = f(a) + f'(a)(x - a)$, where $f'(a) = \left.\frac{dy}{dx}\right|_{x=a}$ represents the slope of the tangent line.

Now we consider the tangent line of parametric equations given by $x(t)$ and $y(t)$. Unlike for explicit function $y = f(x)$ where there can only be one tangent line at a point, it is possible to have more than one tangent line at a point on a parametric curve.

Derivative for Parametric Equations

Consider $y = f(x)$. Applying Chain Rule, we have $\frac{dy}{dt} = f'(x)\frac{dx}{dt} = \frac{dy}{dx}\frac{dx}{dt}$ and hence $\frac{dy}{dx} = \frac{\frac{dy}{dt}}{\frac{dx}{dt}}$, provided $\frac{dx}{dt} \neq 0$.

Similarly, for $x = f(y)$, we have $\frac{dx}{dy} = \frac{\frac{dx}{dt}}{\frac{d}{dt}}$, provided $\frac{dy}{dt} \neq 0$.

This example shows that with parametric curves, it is possible to have multiple tangent lines at a point.

Example

Find the tangent line(s) to the parametric curve given by $x = t^3 - 2t$ amd $y = t^2$ at the point $(0, 2)$.

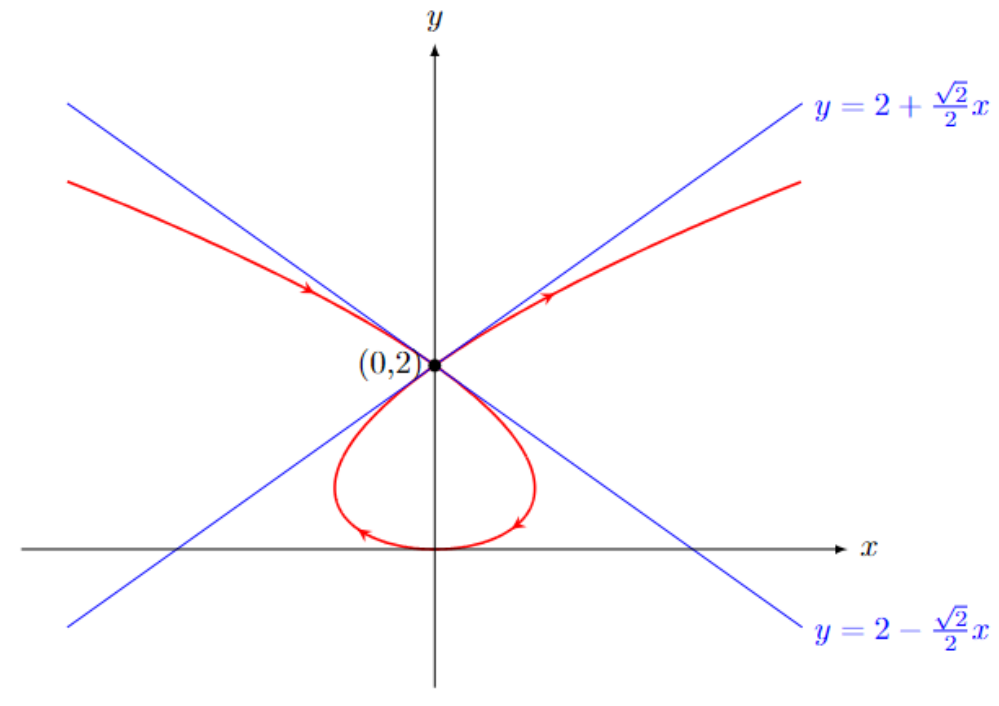
Solution Slope of the tangent line is $m = \dfrac{dy}{dx} = \dfrac{\frac{dx}{dt}}{\frac{d}{dt}} = \dfrac{2t}{3t^2-2}$. Convert the point $(x, y) = (0, 2)$ in terms of t: $0 = t^3 - 2t$ gives $t = 0$ or $t = \pm\sqrt{2}$ while $2 = t^2$ gives $t = \pm\sqrt{2}$.

The value t that satisfy both are $t = \pm\sqrt{2}$, hence there are two tangent lines at the point $(x, y) = (0, 2)$, each correspond to slope at $t = \pm\sqrt{2}$.

Find each slope m and use the slope-intercept form $y = b + mx$ we have the equation of the tangent lines:

- $t = -\sqrt{2} \Rightarrow m = \left.\dfrac{dy}{dx}\right|_{t=-\sqrt{2}} = -\dfrac{2\sqrt{2}}{4} = -\dfrac{\sqrt{2}}{2} \Rightarrow y = 2 - \dfrac{\sqrt{2}}{2}x$
- $t = \sqrt{2} \Rightarrow m = \left.\dfrac{dy}{dx}\right|_{t=\sqrt{2}} = \dfrac{2\sqrt{2}}{4} = \dfrac{\sqrt{2}}{2} \Rightarrow y = 2 + \dfrac{\sqrt{2}}{2}x$

Graphically on the parametric curve:



Horizontal Tangents for Parametric Equations occurs when $\dfrac{dy}{dt} = 0$, provided $\dfrac{dx}{dt} \neq 0$.

Vertical Tangents for Parametric Equations occurs when $\dfrac{dx}{dt} = 0$, provided $\dfrac{dy}{dt} \neq 0$.

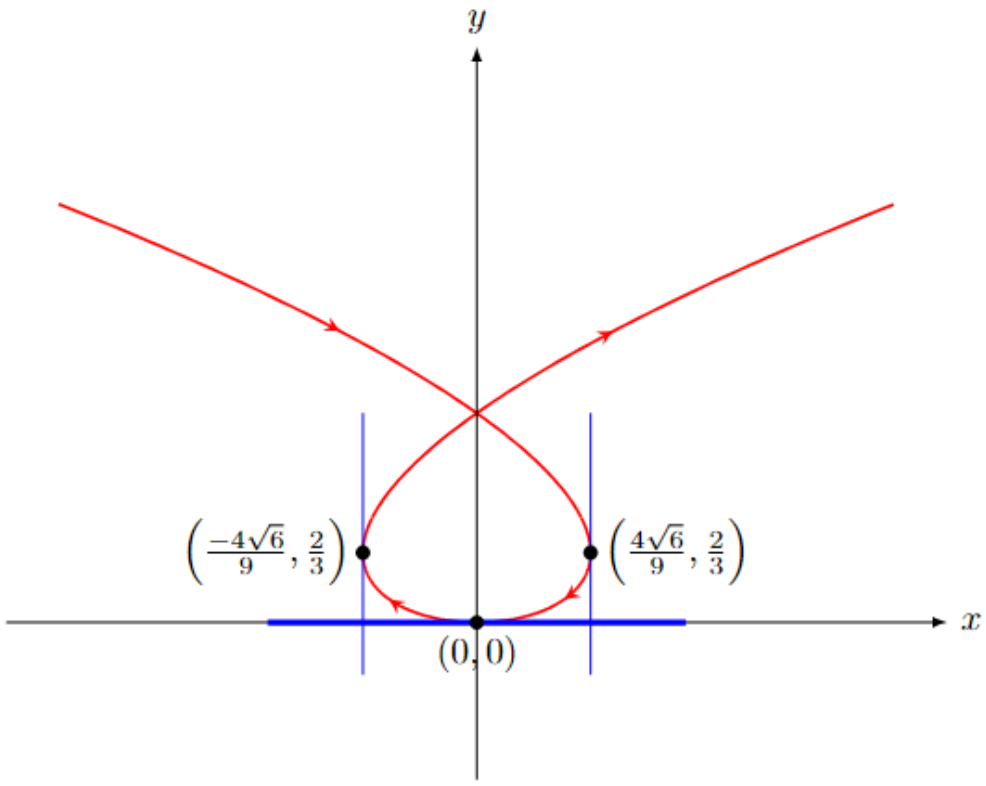
Example

Using the previous example $x(t) = t^3 - 2t$ and $y(t) = t^2$, find the (x, y)-coordinates of the points where the parametric curve has horizontal or vertical tangents.

First derivative of parametric equations: $x(t) = t^3 - 2t \Rightarrow \dfrac{dx}{dt} = 3t^2 - 2$ and $y(t) = t^2 \Rightarrow \dfrac{dy}{dt} = 2t$.

- For Horizontal tangent: $\dfrac{dy}{dt} = 2t \Rightarrow t = 0$. Hence, there is one horizontal tangent, at $(0, 0)$.

- For Vertical tangent: $\frac{dx}{dt} = 3t^2 - 2 \Rightarrow t = \pm\sqrt{\frac{2}{3}}$. Hence, there are two vertical tangents, at $\left(\pm\frac{4\sqrt{6}}{9}, \frac{2}{3}\right)$.



Using the code to find the t values can be done by setting the derivative to zero as follows:

```
Horizontal Tangent at t = [0]
Vertical Tangent at t = [-sqrt(6)/3, sqrt(6)/3]
```

Second Derivative for Parametric Equations

From the first derivative, we can get the second derivative: $\frac{d^2y}{dx^2} = \frac{d}{dx}\left(\frac{dy}{dx}\right) = \frac{\frac{d}{dt}\left(\frac{dy}{dx}\right)}{\frac{dx}{dt}}$.

Example

Using the previous example $x(t) = t^3 - 2t$ and $y(t) = t^2$, find $\frac{d^2y}{dx^2}$.

From first derivative: $\frac{dx}{dt} = 3t^2 - 2 \qquad \frac{dy}{dt} = 2t \qquad \frac{dy}{dx} = \frac{2t}{3t^2 - 2}$.

Hence, $\frac{d}{dt}\left(\frac{dy}{dx}\right) = \frac{d}{dt}\left(\frac{2t}{3t^2 - 2}\right) = \frac{-4 - 6t^2}{(3t^2 - 2)^2}$.

Thus $\frac{d^2y}{dx^2} = \frac{\frac{d}{dt}\left(\frac{dy}{dx}\right)}{\frac{dx}{dt}} = \frac{\frac{-4-6t^2}{(3t^2-2)^2}}{3t^2 - 2} = \frac{-4 - 6t^2}{(3t^2 - 2)^3}$.

8.6.2 Parametric Equations - Area and Arc Length

Area with Parametric Equations

Given the parametric equations $x(t)$ and $y(t)$.

If $x(t)$ is differentiable and the corresponding parametric curve is not self-intersecting on $a \leq t \leq b$, then the area under the parametric curve on the interval is given by $A = \int_a^b y(t) \cdot x'(t)\, dt$.

Example

Find the area under the parametric curve given by $x(t) = t - \sin(t)$ and $y(t) = 1 - \cos(t)$ on $0 \leq t \leq 2\pi$.

It is left to the reader to check that the parametric curve is not self-intersecting (plot the curve!)

Directly using the arc formula we have $A = \int_a^b y(t) \cdot x'(t)\, dt = A = \int_0^{2\pi} (1 - \cos(t))(1 - \cos(t))\, dt = 3\pi$.

The code below ask user for $x(t), y(t), a, b$, and return the area.

```
The area under the curve from t=0 to t=2*pi is: 3*pi
```

Arc Length with Parametric Equations

Given the parametric equations $x(t)$ and $y(t)$ on $a \leq t \leq b$, if $x(t)$ and $y(t)$ are differentiable, and the parametric curve is not self-overlapping, then the arc length of this curve is given by $L = \int_a^b \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2}\, dt$.

Example

Find the area under the parametric curve given by $x(t) = t - \sin(t)$ and $y(t) = 1 - \cos(t)$ on $0 \leq t \leq 2\pi$.

It is left to the reader to check that the parametric curve is not self-overlapping (plot the curve and try to alter the interval!)

Directly using the are formula we have $L = \int_a^b \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2}\, dt = \int_0^{2\pi} \sqrt{(1 - \cos(t))^2 + (\sin(t))^2}\, dt = 8$.

The code below ask user for $x(t), y(t), a, b$, and return the arc length.

```
Arc length of the parametric curve on the interval (0, 2*pi): 8.0
```

Example where curve self-overlapping

Now suppose the problem is finding the arc length for $x(t) = \sin(3t)$ and $y(t) = \cos(3t)$ for $0 \leq t \leq 2\pi$.

Recall that $x(t) = \sin(t)$ and $y(t) = \cos(t)$ for $0 \leq t \leq 2\pi$ gives the unit circle.

Hence $x(t) = \sin(3t)$ and $y(t) = \cos(3t)$ for $0 \leq t \leq 2\pi$ is the unit circle repeated 3 times (check this!)

Thus for the arc length, if we want to use the formula for L, then we must change the b from 2π into $\frac{2\pi}{3}$ to get the circumference of the unit circle:

$$L = \int_0^{\frac{2\pi}{3}} \sqrt{(3\cos(3t))^2 + (-3\sin(3t))^2}\, dt = 2\pi.$$

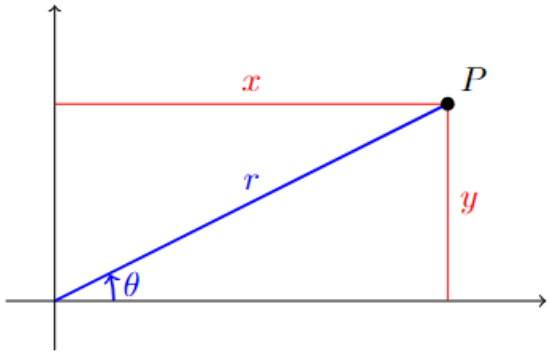
8.6.3 Polar Coordinates - Conversion and Tangents

In the **Cartesian** coordinate system, a point P is given by the coordinate (x, y) when it is moving x units horizontally and y units vertically from the origin $(0, 0)$.

In the **Polar** coordinate system, a point is given by the coordinate (r, θ) when it is distanced r straight from the origin $(0, 0)$, creating a θ angle.

The r in polar coordinates is called the **radial coordinate** or **radian distance** or **radius**. Since it is a distance, r **is nonnegative**.

The θ in polar coordinate is called the **angular coordinate** or **polar angle** or **azimuth**.



Polar to Cartesian Coordinates Conversion

From the diagram above, it is easy to see that $x = r\cos(\theta)$ and $y = r\sin(\theta)$.

Example

Convert $(r, \theta) = \left(4, \frac{\pi}{3}\right)$ into Cartesian coordinates.

- $x = r\cos(\theta) = 4\cos\left(\frac{\pi}{3}\right) = 4 \cdot \frac{1}{2} = 2.$
- $y = r\sin(\theta) = 4\sin\left(\frac{\pi}{3}\right) = 4 \cdot \frac{\sqrt{3}}{2} = 2\sqrt{3}.$

Hence the Cartesian coordinates of $(r, \theta) = \left(4, \frac{\pi}{3}\right)$ is $(x, y) = \left(2, 2\sqrt{3}\right)$.

Below is the code that converts given Polar coordinates into Cartesian coordinates. To lower the risk of a typo involving π in the polar angle θ, the code is set to ask the user for θ to be written in degrees, and it will convert it into radians. Recall that π radians is $180°$.

```
Cartesian coordinates (x, y): (2.0000000000000004, 3.4641016151377544)
```

Cartesian to Polar Coordinates Conversion

Again from the diagram above, by the Pythagorean theorem, $r^2 = x^2 + y^2$. Since r is nonnegative, we have $r = \sqrt{x^2 + y^2}$.

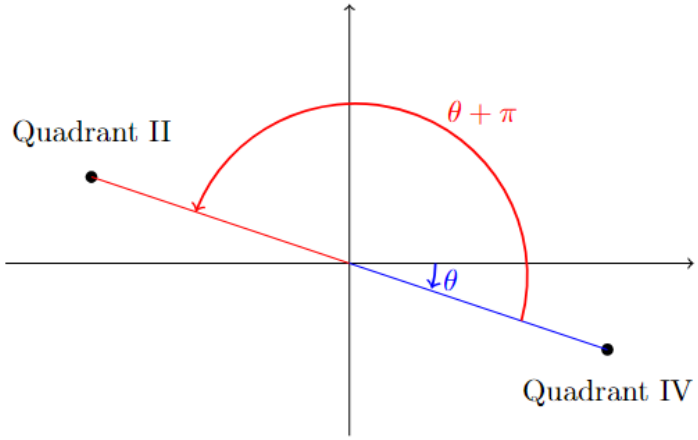
Also, since $\tan(\theta) = \frac{y}{x}$, we have $\theta = \mathbf{arctan}\left(\frac{y}{x}\right)$. However, it is important to **pay attention to the quadrant** in which the coordinate is located.

Example

Convert $(x, y) = (-8, 1)$ into Polar coordinates.

- $r = \sqrt{x^2 + y^2} = \sqrt{(-8)^2 + 1^2} = \sqrt{65} \approx 8.0623$.
- $\theta = \arctan\left(\frac{1}{-8}\right) \approx -0.1244$.
  - However, note that the point $(x, y) = (-8, 1)$ is on the second quadrant, while $-\frac{\pi}{2} < -0.1244 < 0$ means the angle -0.1244 is on the fourth quadrant.
  - To get the correct angle on the second quadrant, $\theta \approx -0.1244 + \pi \approx 3.0172$.

Hence, the Polar coordinate of $(x, y) = (-8, 1)$ is $(r, \theta) = (8.0623, 3.0172)$.



Below is the code that converts given Cartesian coordinates into Polar coordinates. Since we need to pay attention to the quadrant, it is important to set the code using **math.atan2** instead of **math.atan**, as **atan** only return angle in $(-\frac{\pi}{2}, \frac{\pi}{2})$.

```
Polar coordinates (r, theta): (8.06225774829855, 3.017237659043032)
```

Equations Conversion

Using the same conversions, we can convert equations between Polar and Cartesian coordinate systems.

Example

- Convert $x^2 = 4xy - y^2 + 2$ into Polar coordinates

$$\begin{aligned} x^2 &= 4xy - y^2 + 2 \\ (r\cos(\theta))^2 &= 4(r\cos(\theta))(r\sin(\theta)) - (r\sin(\theta))^2 + 2 \\ r^2\left(\cos^2(\theta) + \sin^2(\theta)\right) &= 4r^2\cos(\theta)\sin(\theta) \\ r^2 &= 2r^2\sin(2\theta) \\ 0 &= r^2\left(2\,\mathbf{sin}(2\theta) - 1\right) + 2 \end{aligned}$$

  There is no rule on how far one should "simplify" their equations. It should be okay to leave it as $(r\cos(\theta))^2 = 4(r\cos(\theta))(r\sin(\theta)) - (r\sin(\theta))^2 + 2$. For further purposes, many resources prefer to simplify until the $r = f(\theta)$ form whenever possible.

- Convert $\dfrac{2}{r} = \sin(\theta) - \sec(\theta)$

$$\begin{aligned} \frac{2}{r} &= \sin(\theta) - \sec(\theta) \\ 2 &= r\sin(\theta) - \frac{r}{\cos(\theta)} \\ 2 &= r\sin(\theta) - \frac{r^2}{r\cos(\theta)} \\ 2 &= y - \frac{x^2 + y^2}{x} \end{aligned}$$

  One can also simplify further to $2x = y - x^2 - y^2$.

Tangents with Polar Coordinates

Given the equation in the polar form of $r = f(\theta)$. By applying Cartesian-Polar conversion and Product Rule, we have:

- $x = r\cos(\theta) = f(\theta)\cos(\theta) \Rightarrow \dfrac{dx}{d\theta} = f'(\theta)\cos(\theta) - f(\theta)\sin(\theta) = \dfrac{dr}{d\theta}\cos(\theta) - r\sin(\theta)$.
- $y = r\sin(\theta) = f(\theta)\sin(\theta) \Rightarrow \dfrac{dx}{d\theta} = f'(\theta)\sin(\theta) + f(\theta)\cos(\theta) = \dfrac{dr}{d\theta}\sin(\theta) + r\cos(\theta)$.

Hence, the slope of the tangent line is $m = \dfrac{dy}{dx} = \dfrac{\frac{dr}{d\theta}\,\mathbf{sin}(\theta) + r\,\mathbf{cos}(\theta)}{\frac{dr}{d\theta}\,\mathbf{cos}(\theta) - r\,\mathbf{sin}(\theta)}$.

The equation of the tangent line can then be obtained using the point-slope form $y - y_1 = m(x - x_1)$.

Example

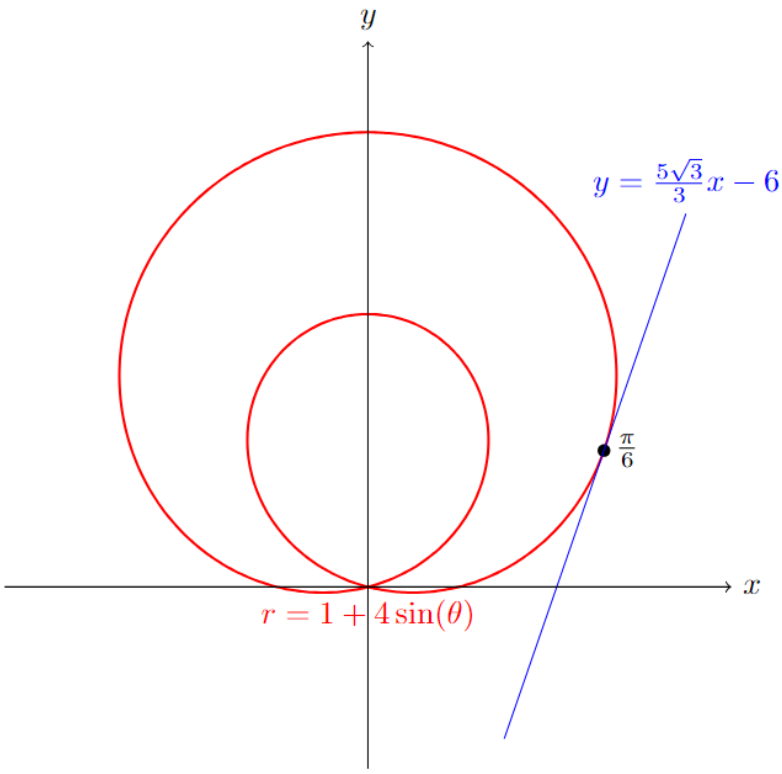
Determine the equation of the tangent line to $r = 1 + 4\sin(\theta)$ at $\theta = \dfrac{\pi}{6}$.

Now $\dfrac{dr}{d\theta} = 4\cos(\theta)$, so $\dfrac{dy}{dx} = \dfrac{(4\cos(\theta))\sin(\theta) + (1 + 4\sin(\theta))\cos(\theta)}{(4\cos(\theta))\cos(\theta) - (1 + 4\sin(\theta))\sin(\theta)}$.

The slope of the tangent line is $m = \left.\dfrac{dy}{dx}\right|_{\theta=\frac{\pi}{6}} = \dfrac{4\cdot\frac{\sqrt{3}}{2}\cdot\frac{1}{2} + \left(1 + 4\cdot\frac{1}{2}\right)\cdot\frac{\sqrt{3}}{2}}{4\cdot\frac{\sqrt{3}}{2}\cdot\frac{\sqrt{3}}{2} - \left(1 + 4\cdot\frac{1}{2}\right)\cdot\frac{1}{2}} = \dfrac{5\sqrt{3}}{3}$.

To use the point slope form, we have $x_1 = r\left(\frac{\pi}{6}\right)\cos\left(\frac{\pi}{6}\right) = \frac{3\sqrt{3}}{2}$ and $y_1 = r\left(\frac{\pi}{6}\right)\sin\left(\frac{\pi}{6}\right) = \frac{3}{2}$.

Hence the equation of the tangent line in point-slope form is $y - \frac{3}{2} = \frac{5\sqrt{3}}{3}\left(x - \frac{3\sqrt{3}}{2}\right)$, or, in slope-intercept form is $y = \frac{5\sqrt{3}}{3}x - 6$.



Below is a code that asks for the polar equation $r(t)$ and the point $t = c$. It returns the equation of the tangent line to $r(t)$ at $t = c$.

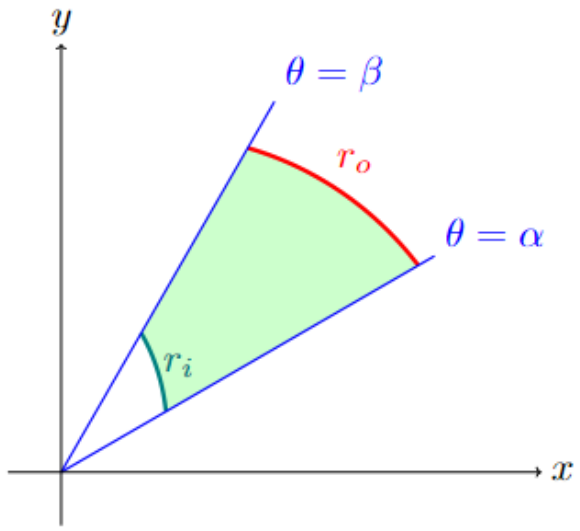
In this instance, t is used instead of θ only for naming convenience, as it is not to be confused with the use of t in the parametric equation.

```
Equation of the tangent line at t = pi/6: y = 5*sqrt(3)/3*x + -6.00000000000000
```

8.6.4 Polar Coordinates - Area and Arc Length

Area with Polar Coordinates

Area with polar coordinates normally refers to the area formed between two polar angles $\theta = \alpha$ and $\theta = \beta$ and some radius(es).

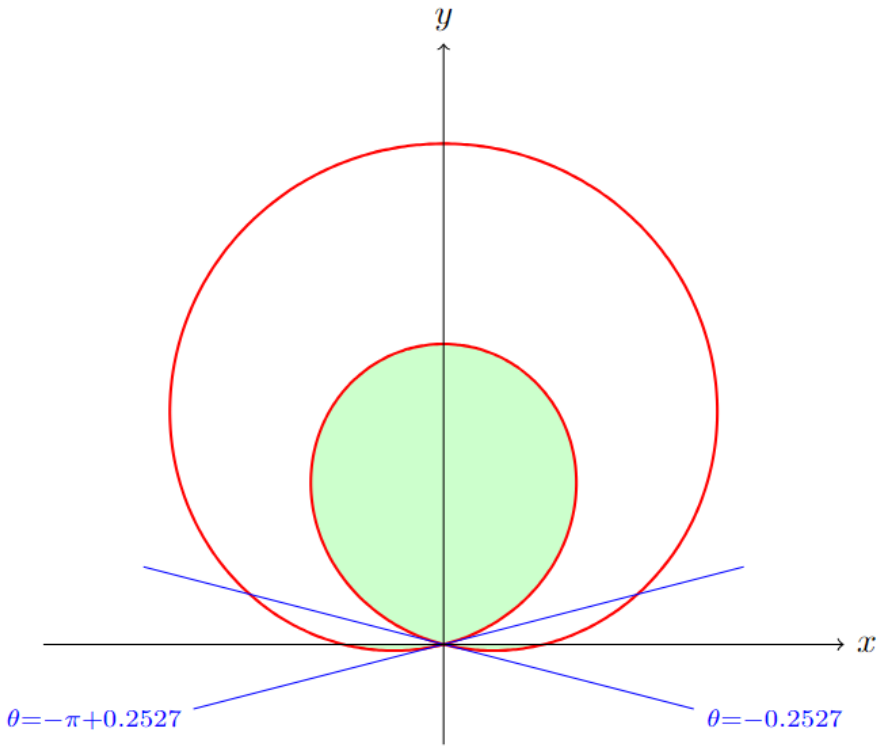


The area is given by $A = \int_{\alpha}^{\beta} \frac{1}{2}\left(r_o^2 - r_i^2\right)\, d\theta$. Here $r_o(\theta)$ and $r_i(\theta)$ is the radius equation that lies on the outside and inside between $\theta = \alpha$ and $\theta = \beta$, respectively.

If there is only one radius equation, then the formula is easily altered by setting $r_i(\theta) = 0$.

Example

Find the area inside the inner loop of $r = 1 + 4\sin(\theta)$.



Since it is a loop, then $r_o = r = 1 + 4\sin(\theta)$ and $r_i = 0$. Since the inner loop goes through the origin, we can find θ by solving $r = 0$.

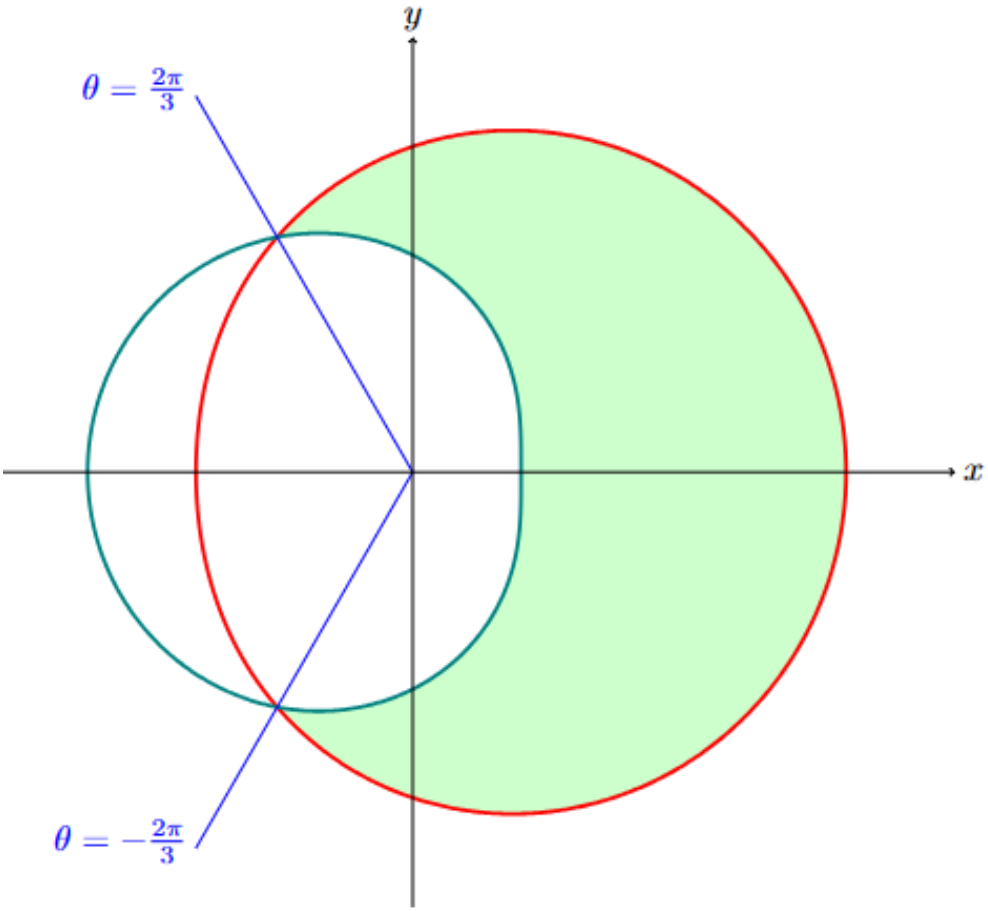
Setting $1 + 4\sin(\theta) = 0$ gives $\theta = \arcsin\left(-\frac{1}{4}\right) \approx -0.2527$ and angle in the fourth quadrant. To trace the inner loop,

trace θ from the third to fourth quadrant.

Hence $\alpha = -\pi + 0.2527$ and $\beta = -0.2527$. The area of the inner loop is then $A = \int_{-\pi+0.2527}^{-0.2527} \frac{1}{2}\left(r = 1 + 4\sin(\theta)\right)^2 \, d\theta \approx 6.39124$.

Example

Find the area inside $r = 6 + 2\cos(\theta)$ but outside $r = 4 - 2\cos(\theta)$.



To find θ, find where the loop intersects: $6 + 2\cos(\theta) = 4 - 2\cos(\theta) \Rightarrow \cos(\theta) = -\frac{1}{2} \Rightarrow \theta = \arccos\left(\frac{1}{2}\right) = \frac{2\pi}{3}$.

To trace the area counterclockwise, we trace θ from the third to second quadrant hence we have $\alpha = -\frac{2\pi}{3}$ and $\beta = \frac{2\pi}{3}$.

The area is then $A = \int_{-\frac{2\pi}{3}}^{\frac{2\pi}{3}} \frac{1}{2}\left[\left(6 + 2\cos(\theta)\right)^2 - \left(4 - 2\cos(\theta)\right)^2\right] \, d\theta \approx 76.5289$.

Arc Length with Polar Coordinates

For then non-overlapping curve of a polar equation $r = f(\theta)$ from $\theta = \alpha$ to $\theta = \beta$, the arc length is given by $L = \int_{\alpha}^{\beta} \sqrt{r^2 + \left(\frac{dr}{d\theta}\right)^2}\, d\theta$.

Example

Determine the arc length of $r = 2\sin(\theta)$ for $0 \leq \theta \leq \pi$.

First, check that the curve is not self-overlapping on the given range of θ. One can plot to check that the curve is a nonoverlapping circle.

Hence, the arc length is

$$\begin{aligned} L &= \int_{\alpha}^{\beta} \sqrt{r^2 + \left(\frac{dr}{d\theta}\right)^2}\, d\theta \\ &= \int_{0}^{\pi} \sqrt{(2\sin(\theta))^2 + (2\cos(\theta))^2}\, d\theta \\ &= \int_{0}^{\pi} \sqrt{4\left(\sin^2(\theta) + \cos^2(\theta)\right)}\, d\theta \\ &= \int_{0}^{\pi} \sqrt{4}\, d\theta = \int_{0}^{\pi} 2\, d\theta = 2\pi \end{aligned}$$

Below is the code that returns the arc length given polar equation $r(t)$ and $\alpha \leq t \leq \beta$. Note that the use of t instead of θ in the code below is only for naming convenience, not to be confused with a parametric equation.

```
Arc length: 2*pi
```

8.6.5 Exercises

Exercises

1. (Section 6.1) Show that for a circle centered at $(h, k) \neq (0, 0)$, the parametric equation is $x = h + r\cos(t)$ and $y = k + r\sin(t)$.
2. (Section 6.1) Sketch the following parametric equations. What shape is the curve?

   If the code fails to recognize π, you can approximate using 3.14.

   - $x = \cos(t)$ and $y = \sin(t)$ for $0 \leq t \leq 2\pi$.
   - $x = 2e^t$ and $y = cos\left(1 + e^{2t}\right)$ for $0 \leq t \leq \frac{2}{3}$
   - $x = \sin\left(e^{\cos(t)} - 2\cos(4t) - \sin^5\left(\frac{t}{12}\right)\right)$ and $y = \cos\left(e^{\cos(t)} - 2\cos(4t) - \sin^5\left(\frac{t}{12}\right)\right)$ for $0 \leq t \leq 20$.
3. (Section 6.1) Given the parametric equations $x(t) = t^2 - 2t - 11$ and $y(t) = -2t^4 + 12t^3 - 64t + 7$.
   - Find the tangent line(s) at the point $(-3, 7)$.

- Find the (x, y)-coordinates of the horizontal and vertical tangents.
- Find $\dfrac{d^2y}{dx^2}$.

4. (Section 6.2) Given the parametric equations $x(t) = 3 - \cos^3(t)$ and $y(t) = 4 + \sin(t)$.
   - Find the area below the parametric curve on $0 \leq t \leq \pi$.
   - Find the arc length of the parametric curve on $0 \leq t \leq 4\pi$.
5. (Section 6.3) Do the following problems:
   - Convert the point $(r, \theta) = \left(-2, \dfrac{2\pi}{3}\right)$ to Cartesian coordinates.
   - Convert the point $(x, y) = (-2\sqrt{3}, -6)$ to Polar coordinates.
   - Convert the equation $6r^3 \sin(\theta) + \cos(\theta) = 4$ to Cartesian coordinates.
   - Convert the equation $6 - xy = \dfrac{2x}{5x^2 + 5y^2}$ to Polar coordinates.
6. (Section 6.3) Determine the equation of the tangent line to $r = \cos(\theta)\sin(4\theta)$ at $\theta = \dfrac{\pi}{6}$.
7. (Section 6.4) Find the area that is inside both $r = 2 + \sin(\theta)$ and $r = 1 - \sin(\theta)$.
   Hint: Wanted area = Area1 − Area2. Figure out what Area1 and Area2 are, and calculate each using the Area formula.
8. (Section 6.4) Determine the arc length of the curve for each given polar equation on each given θ range.
   - $r = \theta$ for $0 \leq \theta \leq 2\pi$
   - $r = \cos(\theta) + \sin(3\theta)$ for $0 \leq \theta \leq \dfrac{2\pi}{3}$

8.7 Sequences and Series

Tip

Below is the list of topics that are covered in this section:

- Sequences
- Series
- Convergence and Divergence of Series
- Convergence Tests
- Power Series

```
import matplotlib.pyplot as plt
import math
```

8.7.1 Sequences

In this module, we define a **sequence** as an arrangement of an infinite number of numbers written in a specific order.

The common notations used for a sequence are $\{a_n\} = \{a_n\}_{n=1}^{\infty} = \{a_1, a_2, ...\}$, where a_i denote the ith term of the sequence.

Example

Write the first 5 terms of the following sequences:

- $\left\{\frac{1}{n}\right\} = \frac{1}{1}, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \ldots$
- $\{(-1)^n\}_{n=0}^{\infty} = 1, -1, 1, -1, 1, \ldots$

Some common sequences:

- **Arithmetic sequence**: created by adding a certain constant d to the preceding term.
  - Using $a_1 = a$, the sequence is written explicitly: $\{a, a+d, a+2d, a+3d, ...\}$
  - Or, written recursively: $\{a_n\}$ where $a_1 = a$ and $a_i = a_{i-1} + d$ for $i = 2, 3, ...$
- **Geometric sequence**: created by multiplying a certain constant ratio r to the preceding term.
  - Using $a_1 = a$, the sequence is written explicitly: $\{a, ar, ar^2, ar^3, ...\}$
  - Or, written recursively: $\{a_n\}$ where $a_1 = a$ and $a_i = a_{i-1} \cdot r$ for $i = 2, 3, ...$
- **Harmonic sequence**: created by taking the reciprocal of an arithmetic sequence.
  - Using $a_1 = \frac{1}{a}$, the sequence is written explicitly: $\left\{\frac{1}{a}, \frac{1}{a+d}, \frac{1}{a+2d}, \frac{1}{a+3d}, \ldots\right\}$
- **Fibonacci sequence**: created by taking each term as the sum of the two preceding terms.
  - Written recursively: $\{a_n\}$ where $a_1 = 0, a_2 = 1$, and $a_i = a_{i-1} + a_{i-2}$ for $i = 3, 4, ...$

Properties of Sequences

Given a sequence $\{a_n\}$.

- $\{a_n\}$ is **increasing** if $a_n < a_{n+1}$ for every n, also called **monotonic increasing**.
- $\{a_n\}$ is **decreasing** if $a_n > a_{n+1}$ for every n, also called **monotonic decreasing**.
- $\{a_n\}$ is **monotonic nonincreasing (or monotonic nondecreasing)** if $\{a_n\}$ never increase (or decrease).
- $\{a_n\}$ is **bounded below** if there exist a number m such that $m \leq a_n$ for every n. Here m is called a **lower bound**.
- $\{a_n\}$ is **bounded above** if there exist a number M such that $M \geq a_n$ for every n. Here M is called a **upper bound**.
- $\{a_n\}$ is **bounded** if it is **both** bounded below and bounded above.

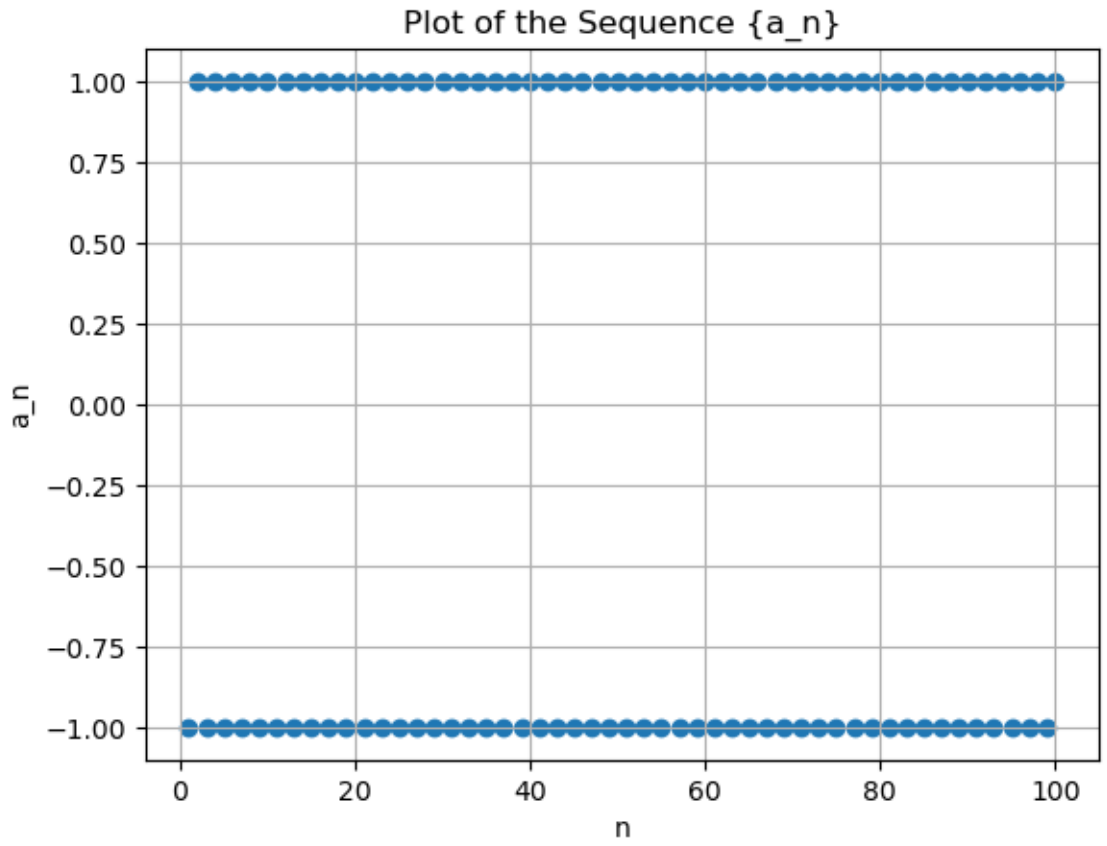
Example

Determine if the given sequence is monotonic and/or bounded.

- $\{a_n\} = \left\{\frac{1}{n}\right\}$

  - $\{a_n\}$ **is monotonic (decreasing)** as $\dfrac{1}{n} > \dfrac{1}{n+1}$ for every n.
  - $\{a_n\}$ **is bounded** since $0 \leq a_n \leq 1$ for every n. The first term is 1, value kept decreasing, but no matter how big n is, $\dfrac{1}{n}$ is never negative.
- $\{a_n\} = \left\{(-1)^n\right\}_{n=0}^{\infty}$
  - $\{a_n\}$ **is NOT monotonic** since the sequence terms alternate between 1 and -1 so neither an increasing nor a decreasing sequence.
  - $\{a_n\}$ **is bounded** since $-1 \leq a_n \leq 1$ for every n.
  - Note that the lower bound and upper bound are not unique. We can also said $\{a_n\}$ **is bounded** since $-35 \leq a_n \leq 708$ for every n.

Below is an example code that plot the first 100 terms of the sequence $\{a_n\}$.



Limit of Sequences

Recall limit notation from the Limit Section.

- We say that $\lim_{n\to\infty} a_n = L$ if for every number $\varepsilon > 0$ there is an integer N such that $|a_n - L| < \varepsilon$ whenever $n > N$.
- We say that $\lim_{n\to\infty} a_n = \infty$ if for every number $M > 0$ there is an integer N such that $a_n > M$ whenever $n > N$.
- We say that $\lim_{n\to\infty} a_n = -\infty$ if for every number $M < 0$ there is an integer N such that $a_n < M$ whenever $n > N$.

If $\lim_{n\to\infty} a_n$ exists and is finite, we say $\{a_n\}$ is **convergent**.

If $\lim_{n\to\infty} a_n$ DNE or is infinite, we say $\{a_n\}$ is **divergent**. If $\lim_{n\to\infty} a_n = \pm\infty$ then we say $\{a_n\}$ diverges to $\pm\infty$.

Note that these limit definitions are not the same with monotonicity, which is stricter.

The use of an integer N whenever $n > N$ emphasizes how limits only consider the end part of a sequence, not necessarily for every n like in monotonicity. For example:

- The sequence $\{a_n\} = \{2, 5, 60, 700, 8000, 43, 521, 1, -70, -500, -2, 1, 1, 1, 1, 1, 1, 1, ...\}$ (the rest of the terms are all 1).>
  - $\{a_n\}$ is not monotonic, but $\lim_{n\to\infty} a_n = 1$ since at some point, all the term goes to 1.
- The sequence $\{a_n\} = \{35, 75, 42, 21, -12, -1, -2, -3, -4, -5, -6, ...\}$ (the rest of the terms keep going down by 1).
  - $\{a_n\}$ is not monotonic decreasing, but $\lim_{n\to\infty} a_n = -\infty$ since at some point the term kept going down.

Properties of limits of sequences follow from the properties of limits, so will not be relisted in this section.

Theorems

Below are some useful theorems on sequences. Motivated students are encouraged to prove them.

Theorems

Theorem 1: If $\{a_n\}$ is monotonic and bounded, then $\{a_n\}$ is convergent.

Theorem 2: If $\lim_{n\to\infty} |a_n| = 0$ then $\lim_{n\to\infty} a_n = 0$.

Theorem 3: The sequence $\{r^n\}_{n=0}^{\infty}$ converges to 0 if $-1 < r < 1$, converges to 1 if $r = 1$, and diverges for all other values of r.

Theorem (Squeeze Theorem for Sequences): If $a_n \le c_n \le b_n$ for all $n > N$ for some integer N and $\lim_{n\to\infty} a_n = \lim_{n\to\infty} b_n = L$, then $\lim_{n\to\infty} c_n = L$.

While the theorems are useful, it is also useful to be able to check the convergence or divergence of a sequence using Python code. Below is a code that checked the first 100 terms of $\{a_n\}$ to determine whether $\{a_n\}$ converges or diverges. It is easy to change the number of terms to check if needed.

```
The sequence diverges.
```

8.7.2 Series

(Infinite) series are an infinite sum of terms $a_1 + a_2 + a_3 + ...$, typically written using sigma notation $\sum_{i=1}^{\infty} a_i$.

Example

- $\sum_{i=1}^{\infty} i = 1 + 2 + 3 + 4 + ...$
- $\sum_{i=1}^{\infty} 2^i = 2^1 + 2^2 + 2^3 + 2^4 + ...$
- $\sum_{i=1}^{\infty} \frac{3^i}{4i} = \frac{3^1}{4 \cdot 1} + \frac{3^2}{4 \cdot 2} + \frac{3^3}{4 \cdot 3} + \frac{3^4}{4 \cdot 4} + ...$

The i in $\sum_{i=1}^{\infty} a_i$ is called the **index (of summation)**, not to be confused with the notation of imaginary numbers.

- What letter used to represent the index does not matter: $\sum_{i=1}^{\infty} a_i = \sum_{k=1}^{\infty} a_k = \sum_{n=1}^{\infty} a_n$, etc.
- For convenience, **index shift** can be applied to series: $\sum_{i=1}^{\infty} a_i = \sum_{i=2}^{\infty} a_{(i-1)} = \sum_{i=0}^{\infty} a_{(i+1)}$, etc.
  - With this in mind, it is normal for different resources and math literatures to define series with index started with $i \neq 1$.

Example

- $\sum_{i=1}^{\infty} \frac{1}{i^2} = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + ... = \sum_{k=1}^{\infty} \frac{1}{k^2}$
- $\sum_{i=1}^{\infty} \frac{1}{i^2} = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + ... = \sum_{i=5}^{\infty} \frac{1}{(i-4)^2}$

8.7.3 Convergence and Divergence of Series

The nth **partial sums** of $\sum_{i=1}^{\infty} a_i$ is the finite sum $S_n = \sum_{i=1}^{n} a_i = a_i + a_2 + ... + a_n$.

The series $\sum_{i=1}^{\infty} a_i$ is **convergent** if the sequence of partial sums $\{S_n\}_{n=1}^{\infty}$ is convergent to a finite limit.

The series $\sum_{i=1}^{\infty} a_i$ is **divergent** if the sequence of partial sums $\{S_n\}_{n=1}^{\infty}$ is divergent.

Example

- $\sum_{i=1}^{\infty} \frac{1}{i^2 + i}$ is convergent since $\lim_{n \to \infty} S_n = \lim_{n \to \infty} \left(\sum_{i=1}^{n} \frac{1}{i^2 + i} \right) = \lim_{n \to \infty} \left(\sum_{i=1}^{n} \frac{1}{i(i+1)} \right) = \lim_{n \to \infty} \left(\sum_{i=1}^{n} \frac{1}{i} - \frac{1}{i+1} \right) = \lim_{n \to \infty} \left(1 - \frac{1}{n} \right) = 1.$

- $\sum_{i=1}^{\infty} i$ is divergent since $\lim_{n\to\infty} S_n = \lim_{n\to\infty}\left(\sum_{i=1}^{n} i\right) = \lim_{n\to\infty}\left(\frac{n(n+1)}{2}\right) = \infty.$

Properties of Convergent Series

If $\sum_{i=k}^{\infty} a_i$ and $\sum_{i=k}^{\infty} b_i$ are both convergent, then

- $\sum_{i=k}^{\infty} ca_i = c\sum_{i=k}^{\infty} a_i$ is also convergent for any constant c.
- $\sum_{i=k}^{\infty} a_i \pm \sum_{i=k}^{\infty} b_i = \sum_{i=k}^{\infty} (a_i \pm b_i)$ is also convergent.

Recall distributive property to note that $\left(\sum_{i=1}^{\infty} a_i\right)\left(\sum_{i=1}^{\infty} b_i\right) \neq \sum_{i=1}^{\infty}(a_i b_i)$. In fact, $\left(\sum_{i=1}^{\infty} a_i\right)\left(\sum_{i=1}^{\infty} b_i\right) = \sum_{i=1}^{\infty} c_i$ where $c_i = \sum_{k=1}^{i} a_k b_{(i-k)}$.

No conclusions can be drawn about the convergence or multiplication of series. Moreover, given how complicated the multiplication of infinite series is, people would rather avoid doing it.

Theorems

Theorem: If $\sum_{n=1}^{\infty} a_n$ converges then $\lim_{n\to\infty} a_n = 0$.

Theorem (Divergence Test): If $\lim_{n\to\infty} a_n \neq 0$ then $\sum_{n=1}^{\infty} a_n$ diverges.

Example The series $\sum_{i=1}^{\infty} \frac{3i^2+5}{7i-i^2}$ diverges since $\lim_{n\to\infty} \frac{3n^2+5}{7n-n^2} = -3 \neq 0$.

Geometric Series

A **geometric series** is a series that can be written in the form $\sum_{i=1}^{\infty} ar^{i-1}$, or, with index shift, $\sum_{i=0}^{\infty} ar^i$.

The nth partial sum S_n of a geometric series can be found as follows:

$$\begin{aligned} S_n - rS_n &= (a + ar + ar^2 + ar^3 + ... + ar^{n-1}) - (ar + ar^2 + ar^3 + ... + ar^{n-1} + ar^n) \\ (1-r)S_n &= a - ar^n \\ S_n &= \frac{a(1-r^n)}{1-r} \end{aligned}$$

Taking the limit of the sequence of partial sums:

$$\begin{aligned}\lim_{n\to\infty} S_n &= \lim_{n\to\infty} \frac{a\left(1-r^n\right)}{1-r} \\ &= \lim_{n\to\infty} \left(\frac{a}{1-r} - \frac{ar^n}{1-r}\right) \\ &= \lim_{n\to\infty} \frac{a}{1-r} - \lim_{n\to\infty} \frac{ar^n}{1-r} \\ &= \frac{a}{1-r} - \frac{a}{1-r} \lim_{n\to\infty} r^n\end{aligned}$$

The only term depending on n is r^n, which only provides finite answers $-1 < r \leq 1$. However, $r = 1$ gives division by zero on the $\frac{a}{1-r}$, and the limit only exists (and equals zero) when $-1 < r < 1$ (or, $|r| < 1$). Thus, the limit of the partial sums, as well as the sum of the geometric series, is $\sum_{i=1}^{\infty} ar^{i-1} = \lim_{n\to\infty} S_n = \frac{a}{1-r} - \frac{a}{1-r} \cdot 0 = \frac{a}{1-r}$.

Example

Determine if $\sum_{n=1}^{\infty} \frac{2^{n+2}}{3^{n-1}}$ converge or diverge. If it converges, give the value of the series.

The convergence of this series can be seen by rewriting it in geometric series form and calculating:

$$\sum_{n=1}^{\infty} \frac{2^{n+2}}{3^{n-1}} = \sum_{n=1}^{\infty} \frac{2^3 \cdot 2^{n-1}}{3^{n-1}} = \sum_{n=1}^{\infty} 8\left(\frac{2}{3}\right)^{n-1} = \frac{8}{1-\frac{2}{3}} = \frac{8}{\frac{1}{3}} = 24.$$

Harmonic Series

A **harmonic series** is the series $\sum_{i=1}^{\infty} \frac{1}{i}$. Remember that if $\sum_{n=1}^{\infty} a_n$ converges then $\lim_{n\to\infty} a_n = 0$, but not the other way around. While $\lim_{n\to\infty} \frac{1}{n} = 0$, the Harmonic series is divergent. This will be shown with the Integral Test below.

8.7.4 Convergence Tests

Integral Test

If $f(x)$ is a continuous, positive, and decreasing function on the interval $[k, \infty)$ and that $f(i) = a_i$, then,

- if $\int_k^{\infty} f(x)\,dx$ is convergent then $\sum_{i=k}^{\infty} a_i$ is also convergent.
- if $\int_k^{\infty} f(x)\,dx$ is divergent then $\sum_{i=k}^{\infty} a_i$ is also divergent.

Example: Harmonic Series

Note that $f(x) = \frac{1}{x}$ is a continuous, positive and decreasing function on $[1, \infty)$.

Since $\int_1^{\infty} \frac{1}{x}\,dx = \infty$ and hence is divergent, then by the Integral test, the Harmonic series $\sum_{i=1}^{\infty} \frac{1}{i}$ also divergent.

The p-series Test

If $k > 0$ then then $\sum_{i=k}^{\infty} \frac{1}{i^p}$ converges if $p > 1$ and diverges if $p \leq 1$.

Example

$\sum_{i=2}^{\infty} \frac{1}{i^5}$ converges as it is a p-series with $p = 5 > 1$, while $\sum_{i=2}^{\infty} \frac{1}{\sqrt{i}}$ diverges as it is a p-series with $p = 0.5 < 1$.

Comparison Test

Given two series $\sum a_i$ and $\sum b_i$ with nonnegative a_i and b_i for all i. If $a_i \leq b_i$ for all i, then:

- If $\sum b_i$ is convergent then $\sum a_i$ is also convergent
- If $\sum a_i$ is divergent then $\sum b_i$ is also divergent

Example

- $\sum_{i=1}^{\infty} \frac{i^3+2}{i^7-5}$ is convergent, because $\frac{i^3+2}{i^7-5} > \frac{i^3+2}{i^7} > \frac{i^3}{i^7} = \frac{1}{i^4}$ and $\sum_{i=1}^{\infty} \frac{1}{i^4}$ is convergent as it is a p-series with $p = 4 > 1$.
- $\sum_{i=1}^{\infty} \frac{i}{i^2+7}$ is divergent, because $\frac{i}{i^2+7} < \frac{i}{i^2} = \frac{1}{i}$ and $\sum_{i=1}^{\infty} \frac{1}{i}$ is divergent as it is a p-series with $p = 1$.

Limit Comparison Test

Given two series $\sum a_i$ and $\sum b_i$ with $a_i \geq 0$ and $b_i > 0$ for all i. Define $c = \lim_{i\to\infty} \frac{a_i}{b_i}$.

If $0 < c < \infty$, then either both $\sum a_i$ and $\sum b_i$ converge or both $\sum a_i$ and $\sum b_i$ diverge.

Limit Comparison Test (LCT) is similar to the Comparison Test (CT) in the sense that we want to compare the series with a series that we know or easily recognize as converging or diverging.

Example

Is $\sum_{i=1}^{\infty} \frac{1}{i+2^i}$ convergent or divergent?

Take $\frac{1}{i+2^i}$ as b_i and $\frac{1}{2^i}$ as a_i so that $c = \lim_{i\to\infty} \frac{\frac{1}{2^i}}{\frac{1}{i+2^i}} = \lim_{i\to\infty} \frac{i+2^i}{2^i} = \lim_{i\to\infty} \frac{i}{2^i} + \lim_{i\to\infty} 1 = \lim_{i\to\infty} \frac{1}{2^i \ln(2)} + 1 = 1$.

As $c = 1$ is finite positive and $\sum_{i=1}^{\infty} \frac{1}{2^i}$ is convergent for being a geometric series with $|r| = \left|\frac{1}{2}\right| < 1$, then by LCT, $\sum_{i=1}^{\infty} \frac{1}{i+2^i}$ is also convergent.

Alternating Series Test

Suppose $\sum a_i$ is a series with $a_i = (-1)^i b_i$ or $a_i = (-1)^{i+1} b_i$ where $b_i \geq 0$ for all i.

If $\lim\limits_{i\to\infty} b_i = 0$ and $\{b_i\}$ is a decreasing sequence, then $\sum a_i$ is convergent.

Example

Determine if $\sum\limits_{i=3}^{\infty} \dfrac{\sin(i\pi)}{\sqrt{i}}$ is convergent or divergent.

Note that $\sin(i\pi) = (-1)^i$ hence we can rewrite the series into $\sum\limits_{i=3}^{\infty} \dfrac{(-1)^i}{\sqrt{i}}$, which is in the alternating form with $b_i = \dfrac{1}{\sqrt{i}}$.

Since $\lim\limits_{i\to\infty} b_i = \lim\limits_{i\to\infty} \dfrac{1}{\sqrt{i}} = 0$ and $\dfrac{1}{\sqrt{i}} > \dfrac{1}{\sqrt{i+1}}$ for all $i \geq 3$, then the Alternating Series Test conditions are satisfied and hence $\sum\limits_{i=3}^{\infty} \dfrac{\sin(i\pi)}{\sqrt{i}}$ is convergent.

Absolute and Conditional Convergence

The series $\sum\limits_{i=1}^{\infty} a_i$ is **absolutely convergent** if $\sum\limits_{i=1}^{\infty} a_i$ converges and $\sum\limits_{n=i}^{\infty} |a_i|$ converges.

The series $\sum\limits_{i=1}^{\infty} a_i$ is **conditionally convergent** if $\sum\limits_{i=1}^{\infty} a_i$ converges but $\sum\limits_{n=i}^{\infty} |a_i|$ diverges.

It immediately follows that absolute convergence implies convergence, but not vice versa.

Example

$\sum\limits_{i=1}^{\infty} \dfrac{(-1)^n}{n}$ is conditionally convergent:

- $\sum\limits_{i=1}^{\infty} \dfrac{(-1)^n}{n}$ is convergent by the Alternating Series Test
- $\sum\limits_{i=1}^{\infty} \left|\dfrac{(-1)^n}{n}\right| = \sum\limits_{i=1}^{\infty} \dfrac{1}{n}$ since it's a Harmonic series (by the Integral Test).

Ratio Test

Given the series $\sum a_i$. Define $L = \lim\limits_{i\to\infty} \left|\dfrac{a_{i+1}}{a_i}\right| = \lim\limits_{i\to\infty} \left|a_{i+1} \cdot \dfrac{1}{a_i}\right|$.

- If $L < 1$ then the series is absolutely convergent, and hence convergent
- If $L > 1$ then the series is divergent
- If $L = 1$ then the test is inconclusive

Example

- $\sum_{i=1}^{\infty} \frac{i^2}{(2i)!}$ is absolutely convergent since

$$L = \lim_{i\to\infty} \left| \frac{(i+1)^2}{(2(i+1))!} \cdot \frac{(2i)!}{i^2} \right| = \lim_{i\to\infty} \left| \frac{(i+1)^2(2i)!}{i^2(2i+2)!} \right|$$
$$= \lim_{i\to\infty} \frac{(i+1)^2(2i)!}{i^2(2i+2)(2i+1)(2i!)}$$
$$= \lim_{i\to\infty} \frac{(i+1)^2}{i^2(2i+2)(2i+1)} = 0 < 1.$$

- $\sum_{i=1}^{\infty} \frac{i!}{2^i}$ is divergent since $L = \lim_{i\to\infty} \left| \frac{(i+1)!}{2^{i+1}} \cdot \frac{2^i}{i!} \right| = \lim_{i\to\infty} \frac{i+1}{2} = \infty > 1.$
- Ratio Test fails for determining convergence of $\sum_{i=1}^{\infty} \frac{1}{i^2}$ since $L = \lim_{i\to\infty} \left| \frac{(i+1)^2}{i^2} \right| = \lim_{i\to\infty} \frac{i^2+2i+1}{i^2} = \lim_{i\to\infty} 1 + \frac{2}{i} + \frac{1}{i^2} = 1.$
  - However, we knew from p-test that $\sum_{i=1}^{\infty} \frac{1}{i^2}$ is convergent.

Root Test

Given the series $\sum a_i$. Define $L = \lim_{i\to\infty} \sqrt[i]{|a_i|} = \lim_{i\to\infty} |a_i|^{1/i}$.

- If $L < 1$ then the series is absolutely convergent, and hence convergent
- If $L > 1$ then the series is divergent
- If $L = 1$ then the series is inconclusive

In using the Root Test, we often use the fact that $\lim_{n\to\infty} n^{1/n} = 1$.

Example

- $\sum_{i=1}^{\infty} \left(\frac{3-4i^2}{5i^2+6} \right)^i$ is absolutely convergent since $L = \lim_{i\to\infty} \left| \left(\frac{3-4i^2}{5i^2+6} \right)^i \right|^{\frac{1}{i}} = \lim_{i\to\infty} \left| \frac{3-4i^2}{5i^2+6} \right| = \left| \frac{-4}{5} \right| = \frac{4}{5} < 1.$
- $\sum_{i=1}^{\infty} \frac{2^i}{i}$ is divergent since $L = \lim_{i\to\infty} \left| \frac{2^i}{i} \right|^{\frac{1}{i}} = \lim_{i\to\infty} \frac{2}{i^{1/i}} = \frac{2}{1} > 1.$
- The Root Test fails for determining convergence of $\sum_{i=1}^{\infty} \frac{1}{i}$ since $L = \lim_{i\to\infty} \left| \frac{1}{i} \right|^{\frac{1}{i}} = \lim_{i\to\infty} \frac{1}{i^{1/i}} = \frac{1}{1} = 1.$
  - However, we already know that this Harmonic series is divergent.

Convergence Test with Python

While multiple tests can be used to manually determine the convergence of a series with steps that can become complicated, in Python, testing the convergence of a series is built in.

Below is a code that ask for a_n and return the convergence of $\sum_{n=1}^{\infty}$. The lower bound $n = 1$ can be changed as needed.

```
The series is convergent.
```

8.7.5 Power Series

A **power series (about** a**)** is a series that can be written in the form $\sum_{i=0}^{\infty} c_i(x-a)^i$ where a and c_i are constants. The c_i's are called the **coefficient** of the power series.

Since power series are a function of x, the convergence of the series depends on the values of x.

The **radius of convergence** of a power series is the number R such that the series will converge for $|x-a| < R$ and diverge for $|x-a| > R$. The series may or may not converge if $|x-a| = R$.

The **interval of convergence** is the interval containing all values of x in which a power series converges, hence it's the interval $(a-R, a+R)$ possibly combined with the endpoint $a-R$ and/or $a+R$.

Example

Determine the radius and interval of convergence for $\sum_{i=0}^{\infty} \frac{2^i(x-2)^i}{i}$.

Using Ratio Test: $L = \lim_{i\to\infty} \left| \frac{2^{i+1}(x-2)^{i+1}}{i+1} \cdot \frac{i}{2^i(x-2)^i} \right| = \lim_{i\to\infty} \left| \frac{2i(x-2)}{i+1} \right| = 2|x-2| \lim_{i\to\infty} \frac{i}{i+1} = 2|x-2|.$

By Ratio Test, $\sum_{i=0}^{\infty} \frac{2^i(x-2)^i}{i}$ converges when $2|x-2| < 1 \Rightarrow |x-2| < 0.5$.

Hence the **radius of convergence is 0.5**.

For the interval of convergence, solving for x from $|x-2| < 0.5 \Rightarrow 1.5 < x < 2.5$, so we are left to check for the endpoints:

- at $x = 1.5$, the sum is $\sum_{i=0}^{\infty} \frac{2^i(1.5-2)^i}{i} = \sum_{i=0}^{\infty} \frac{2^i}{i} \left(\frac{-1}{2}\right)^i = \sum_{i=0}^{\infty} \frac{(-1)^i}{i}$, which converges by the Alternating Series Test.
- at $x = 2.5$, the sum is $\sum_{i=0}^{\infty} \frac{2^i(2.5-2)^i}{i} = \sum_{i=0}^{\infty} \frac{2^i}{i} \left(\frac{1}{2}\right)^i = \sum_{i=0}^{\infty} \frac{1}{i}$, which diverges since it is a Harmonic series.

Thus, the **interval of convergence is** $1.5 \leq x < 2.5$.

8.7.6 Exercises

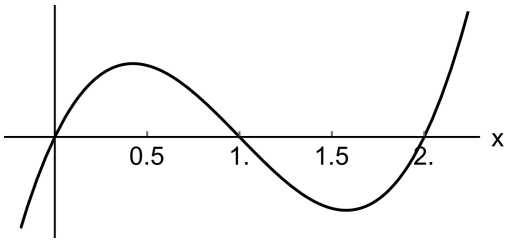
Exercises

1. List the first 5 terms of each given sequence, then determine whether each sequence is monotonic and/or bounded.
   - $\left\{\dfrac{n}{n+1}\right\}_{n=1}^{\infty}$
   - $\left\{\dfrac{2n^2-1}{n}\right\}_{n=2}^{\infty}$
2. Determine whether the following sequences converge or diverge. Determine the limit if it exists.
   - $\left\{\dfrac{(-1)^n \cdot n^3}{5-n^4}\right\}_{n=0}^{\infty}$
   - $\left\{\dfrac{e^{3n}}{2+e^n}\right\}_{n=1}^{\infty}$
   - $\left\{\dfrac{\ln(n+6)}{\ln(3+3n)}\right\}_{n=1}^{\infty}$
3. Index shift and rename exercise:
   - Write $\sum_{i=1}^{\infty} \dfrac{5-i}{i^3}$ as a series that starts at $n=3$.
   - Write $\sum_{i=5}^{\infty} (-1)^{i+2} 2^i$ as a series that starts at $k=1$.
4. Determine the convergence of the following series:
   - $\sum_{i=1}^{\infty} \dfrac{1}{i^{\pi/4}}$
   - $\sum_{i=1}^{\infty} \dfrac{i-1}{\sqrt{i^4+2}}$
   - $\sum_{i=1}^{\infty} \dfrac{(-1)^{i+2}}{i^3+2i+2}$
5. Determine the interval and radius of convergence of the following series:
   - $\sum_{i=1}^{\infty} \dfrac{4^{2i+1}}{5^{1+i}}(x+3)^i$
   - $\sum_{i=1}^{\infty} \dfrac{(i+1)}{(2i+1)!}(x-2)^i$

8.8 Solution to Exercises

8.8.1 Functions

1. Finding domain of functions.
   - Domain of $f(x) = \sqrt{x-9}$ is $[9, \infty)$.
   - Domain of $f(x) = \dfrac{1}{\sqrt{x}-9}$ is $[0, 81) \cup (81, \infty)$.
   - Domain of $f(x) = \ln(x-9)$ is $(9, \infty)$.
2. Given the graph of the function f below.



   - The intervals on which the graph of f is increasing are $(-\infty, 0.5)$ and $(1.5, \infty)$, decreasing is $(0.5, 1.5)$.
   - The intervals on which the graph of f is concave up is $(1, \infty)$, concave down is $(-\infty, 1)$.
   - The only inflection point is at $(1, 0)$.
3. Plot the following functions on the interval $[-5, 5]$ by using the code.
   - $f(x) = 4 + 3x$
   - $f(x) = 4 \cdot 3^x \rightarrow$ increasing fast and can make the other plot unclear, hidden but can uncomment row to show the plot
   - $f(x) = 4 + 3\log(x) \rightarrow$ note that $log(x)$ only defined when $x > 0$, hence the error message
   - $f(x) = 4\cos(3x)$
   - $f(x) = \dfrac{1.5}{1 + 4e^{-2.45x}}$

   (Can always show the plot one-by-one by commenting the row of the other plots)

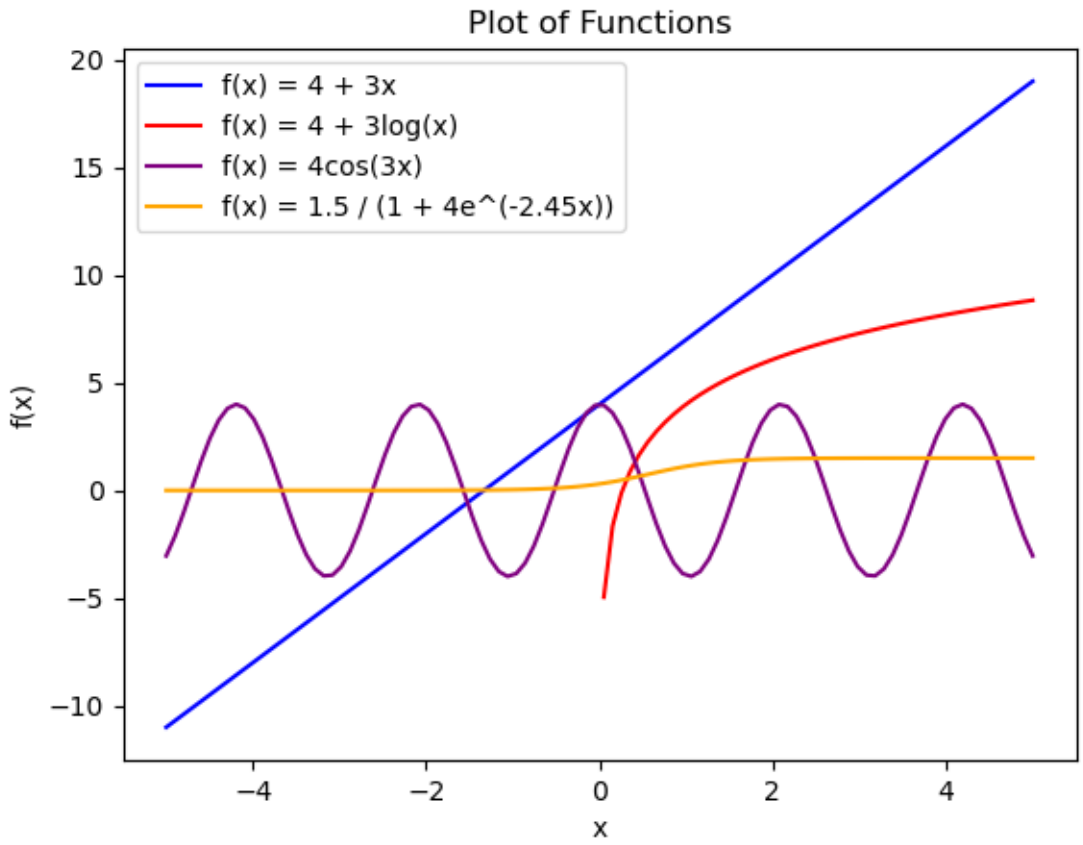
```
import numpy as np
import matplotlib.pyplot as plt

# Define the functions
def f1(x):
    return 4 + 3 * x

def f2(x):
    return 4 * 3**x

def f3(x):
    return 4 + 3 * np.log(x)

def f4(x):
    return 4 * np.cos(3 * x)
```

(continues on next page)

(continued from previous page)

```
def f5(x):
    return 1.5 / (1 + 4 * np.exp(-2.45 * x))

# Define the interval
x = np.linspace(-5, 5, 100)

# Plot the functions
plt.plot(x, f1(x), label='f(x) = 4 + 3x', color='blue')
#plt.plot(x, f2(x), label='f(x) = 4 * 3^x', color='green')  # the function f(2) values increasing fast and can make the other plot unclear. uncomment this row to show.
plt.plot(x, f3(x), label='f(x) = 4 + 3log(x)', color='red') # note that log(x) only defined when x>0, hence the error message.
plt.plot(x, f4(x), label='f(x) = 4cos(3x)', color='purple')
plt.plot(x, f5(x), label='f(x) = 1.5 / (1 + 4e^(-2.45x))', color='orange')

# Set plot title and labels
plt.title('Plot of Functions')
plt.xlabel('x')
plt.ylabel('f(x)')

# Add legend
plt.legend()

# Show the plot
plt.show()
```

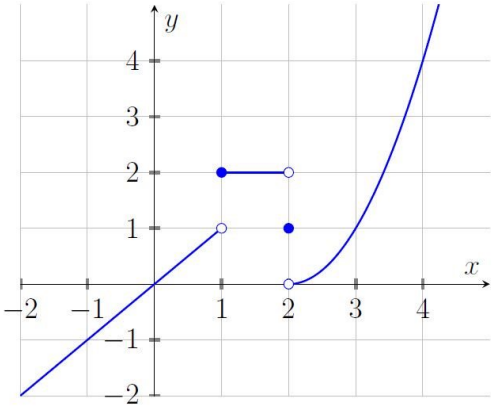
```
C:\Users\isinggih\AppData\Local\Temp\ipykernel_21824\2646948458.py:12: RuntimeWarning: invalid value encountered in log
  return 4 + 3 * np.log(x)
```



4. Given $f(x) = 1 + 6x^2$ and $g(x) = 6 - \sqrt{x}$.
    - $[f + g](1234) \approx 9136507.872$
    - $(g \circ f)(2345) \approx -5738.053534$
    - $[f \cdot g](0.3456) \approx 9.29064512$
    - $(f \circ f)(4567) = 93967528536925351$
    - $g^{-1}(0.2345) = 33.24099025$

8.8.2 Limits, Continuity, and Rates

1. Given the graph of $f(x)$ below:



   - $\lim_{x\to 2} f(x)$ does not exist since left limit $\neq$ right limit
   - $\lim_{x\to 3} f(x)$ exists since left limit = right limit

2. Calculate the following limits. Explain why or why not the limit exists.
   - $\lim_{x\to 7} 5\ln(x) = 5\ln(7) \approx 9.729550745$. The function $\ln(x)$ is continuous on $x > 0$, hence limit exists at $x = 7$.
   - $\lim_{x\to 3} (3x^3 + 4x^2 - 5x - 6) = 96$ The function $3x^3 + 4x^2 - 5x - 6$ continuous on $(-\infty, \infty)$, hence limit exists at $x = 3$.
   - $\lim_{x\to 0} \frac{1}{x}$ does not exists. $\lim_{x\to 0^-} \frac{1}{x} = -\infty$ while $\lim_{x\to 0^+} \frac{1}{x} = \infty$

3. Do the following problems:
   - $\lim_{x\to 0} e^{\sin(x)} = 1$
   - The zeroes of $f(x) = 4x^3 + 3x^2 - 3x + 0.2$ on the interval $[0, 1]$ are $x \approx 0.0724173$ and $x \approx 0.515903$

4. Given $f(x) = 4x^3 - 2x^2 + 3x - 7$.
   - Change of f between $x = -1$ and $x = 2$ is 39
   - Percent change of f between $x = -1$ and $x = 2$ is $\frac{975}{4} = 243.75$ (in %)
   - Average rate of change of f between $x = -1$ and $x = 2$ is 13
   - Instantaneous rate of change of f at $x = 3$ is 99
   - Percent rate of change of f at $x = 3$ is $\frac{2475}{23} \approx 107.608$ (in % per units of x)

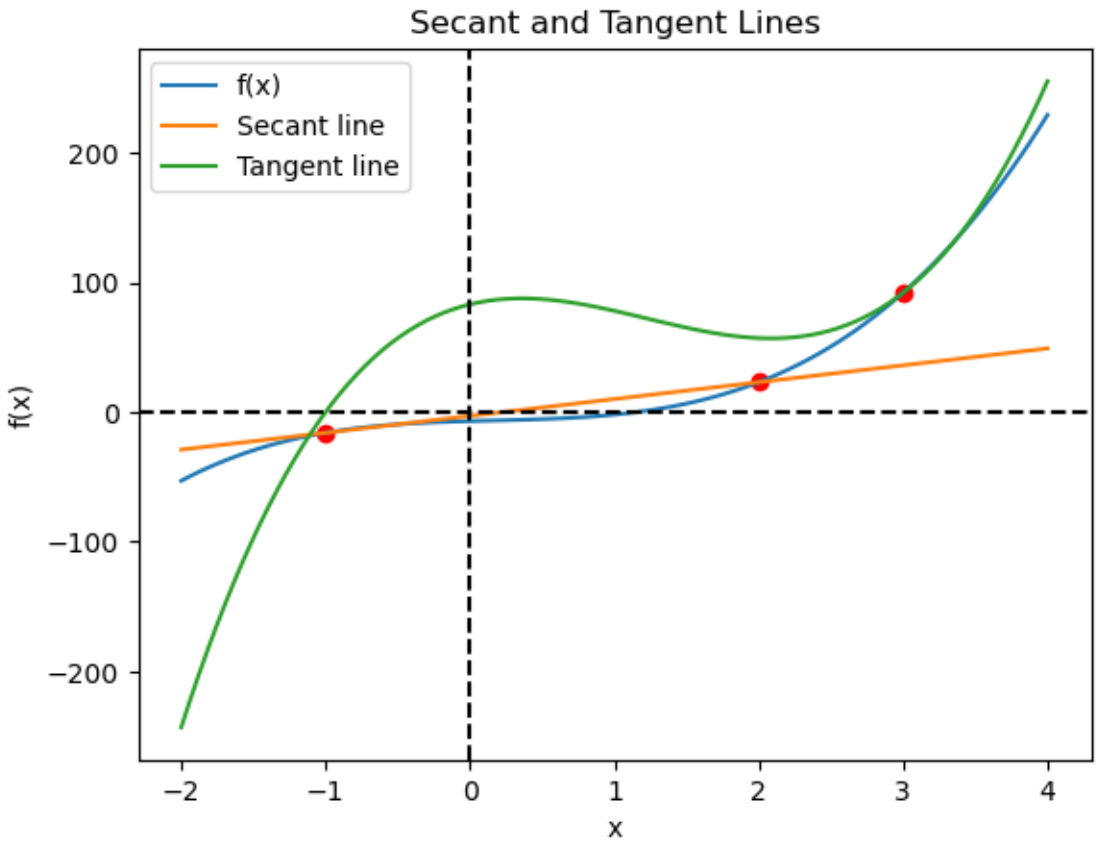
5. Given $f(x) = 4x^3 - 2x^2 + 3x - 7$. Plot the following:
   - Secant line between on f between $x = -1$ and $x = 2$
   - Tangent line on f at $x = 3$

```
import numpy as np
import matplotlib.pyplot as plt

# Define the function
def f(x):
```

(continues on next page)

(continued from previous page)

```
    return 4*x**3 - 2*x**2 + 3*x - 7

# Define the secant line
def secant_line(x):
    return f(-1) + ((f(2) - f(-1)) / (2 - (-1))) * (x - (-1))

# Define the tangent line
def tangent_line(x):
    return f(3) + (4 * 3 * x**2 - 4 * 2 * x + 3) * (x - 3)

# Generate x values
x = np.linspace(-2, 4, 100)

# Generate y values for f(x)
y = f(x)

# Generate y values for the secant line
secant_y = secant_line(x)

# Generate y values for the tangent line
tangent_y = tangent_line(x)

# Plot the function and lines
plt.plot(x, y, label='f(x)')
plt.plot(x, secant_y, label='Secant line')
plt.plot(x, tangent_y, label='Tangent line')

# Mark the points (-1, f(-1)), (2, f(2)), and (3, f(3))
plt.scatter([-1, 2, 3], [f(-1), f(2), f(3)], color='red')

# Add the x=0 and y=0 axes
plt.axhline(0, color='black', linestyle='--')
plt.axvline(0, color='black', linestyle='--')

# Set the x-axis and y-axis labels
plt.xlabel('x')
plt.ylabel('y')

# Set axis labels and title
plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('Secant and Tangent Lines')

# Add legend
plt.legend()

# Show the plot
plt.show()
```



8.8.3 Derivatives

1. Find the derivative of the following functions using the limit definition of the derivative.
   - $f(x) = 3x^2 - 12x + 32$

$$
\begin{aligned}
f'(x) &= \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \\
&= \lim_{h \to 0} \frac{[3(x+h)^2 - 12(x+h) + 32] - [3x^2 - 12x + 32]}{h} \\
&= \lim_{h \to 0} \frac{[3\left(x^2 + 2xh + h^2\right) - 12x - 12h + 32] - [3x^2 - 12x + 32]}{h} \\
&= \lim_{h \to 0} \frac{[3x^2 + 6xh + 3h^2 - 12x - 12h + 32] - [3x^2 - 12x + 32]}{h} \\
&= \lim_{h \to 0} \frac{6xh + 3h^2 - 12h}{h} \\
&= \lim_{h \to 0} (6x + 3h - 12) = 6x - 12
\end{aligned}
$$

- $g(x) = \dfrac{x}{x+2}$

$$\begin{aligned}
g'(x) &= \lim_{h\to 0} \frac{g(x+h)-g(x)}{h} \\
&= \lim_{h\to 0} \frac{\dfrac{x+h}{x+h+2} - \dfrac{x}{x+2}}{h} \\
&= \lim_{h\to 0} \frac{\dfrac{(x+h)(x+2)}{(x+h+2)(x+2)} - \dfrac{x(x+h+2)}{(x+h+2)(x+2)}}{h} \\
&= \lim_{h\to 0} \frac{x^2+2x+hx+2h-x^2-xh-2x}{h(x+h+2)(x+2)} \\
&= \lim_{h\to 0} \frac{2h}{h(x+h+2)(x+2)} \\
&= \lim_{h\to 0} \frac{2}{(x+h+2)(x+2)} = \frac{2}{(x+2)^2}
\end{aligned}$$

- $h(x) = \sqrt{4x^2-6}$

$$\begin{aligned}
h'(x) &= \lim_{h\to 0} \frac{h(x+h)-h(x)}{h} \\
&= \lim_{h\to 0} \frac{\sqrt{4(x+h)^2-6} - \sqrt{4x^2-6}}{h} \cdot \frac{\sqrt{4(x+h)^2-6} + \sqrt{4x^2-6}}{\sqrt{4(x+h)^2-6} + \sqrt{4x^2-6}} \\
&= \lim_{h\to 0} \frac{[4(x+h)^2-6] - [4x^2-6]}{h\left(\sqrt{4(x+h)^2-6} + \sqrt{4x^2-6}\right)} \\
&= \lim_{h\to 0} \frac{4x^2+8xh+4h^2-6-4x^2+6}{h\left(\sqrt{4(x+h)^2-6} + \sqrt{4x^2-6}\right)} \\
&= \lim_{h\to 0} \frac{8xh+4h^2}{h\left(\sqrt{4(x+h)^2-6} + \sqrt{4x^2-6}\right)} \\
&= \lim_{h\to 0} \frac{8x+4h}{\sqrt{4(x+h)^2-6} + \sqrt{4x^2-6}} \\
&= \frac{8x}{\sqrt{4x^2-6} + \sqrt{4x^2-6}} = \frac{4x}{\sqrt{4x^2-6}}
\end{aligned}$$

2. Proof $[f \pm g]' = f' \pm g'$ using the limit definition of the derivative.

$$\begin{aligned}
f' \pm g' &= \lim_{h\to 0} \frac{f(x+h)-f(x)}{h} \pm \lim_{h\to 0} \frac{g(x+h)-g(x)}{h} \\
&= \lim_{h\to 0} \left(\frac{f(x+h)-f(x)}{h} \pm \frac{g(x+h)-g(x)}{h} \right) \\
&= \lim_{h\to 0} \frac{[f(x+h)-f(x)] \pm [g(x+h)-g(x)]}{h} \\
&= \lim_{h\to 0} \frac{f(x+h) \pm g(x+h) - f(x) \mp g(x)}{h} \\
&= \lim_{h\to 0} \frac{[f(x+h) \pm g(x+h)] - [f(x) \pm g(x)]}{h} \\
&= \lim_{h\to 0} \frac{[(f \pm g)(x+h)] - [(f \pm g)(x)]}{h} \\
&= [f \pm g]'
\end{aligned}$$

3. Find the derivative of the following functions.
   - $f(x) = 3x^2 - 12x + 32 \Rightarrow f'(x) = 6x - 12$
   - $g(x) = \dfrac{x}{x+2} \Rightarrow g'(x) = \dfrac{2}{(x+2)^2}$
   - $h(x) = \sqrt{4x^2 - 6} \Rightarrow h'(x) = \dfrac{1}{2}\left(4x^2 - 6\right)^{-1/2} \cdot 8x = \dfrac{4x}{\sqrt{4x^2 - 6}}$
   - $z(x) = x^5 \sin\left(3x^4 + 1\right) + \ln\left(20^x - e^{5x}\right) \Rightarrow z'(x) = 5x^4 \mathbf{sin}\left(3x^4 + 1\right) + x^5 \mathbf{cos}\left(3x^4 + 1\right) \cdot 12x^3 + \dfrac{20^x \mathbf{ln}(20) - 5e^{5x}}{20^x - e^{5x}}$
4. Find the derivative of the following function using both Product and Quotient Rule, and verify that they gives the same answer.
   - $a(x) = \dfrac{\sin(x)}{x} = \sin(x) \cdot x^{-1}$
     - **Product Rule:** $a'(x) = \cos(x) \cdot x^{-1} + \sin(x) \cdot \left(-x^{-2}\right) = \dfrac{\mathbf{cos}(x)}{x} - \dfrac{\mathbf{sin}(x)}{x^2}$
     - **Quotient Rule:** $a'(x) = \dfrac{\cos(x) \cdot x - \sin(x) \cdot 1}{x^2} = \dfrac{\mathbf{cos}(x)}{x} - \dfrac{\mathbf{sin}(x)}{x^2}$
   - $b(x) = \dfrac{\cos(x)}{2 - x^9} = \cos(x) \cdot \left(2 - x^9\right)^{-1}$
     - **Product Rule:** $b'(x) = \left(-\sin(x)\right) \cdot \left(2 - x^9\right)^{-1} + \cos(x) \cdot \left(-1\left(2 - x^9\right)^{-2}\right)\left(-9x^8\right) = \dfrac{-\mathbf{sin}(x)}{2 - x^9} + \dfrac{9x^8 \mathbf{cos}(x)}{\left(2 - x^9\right)^2}$
     - **Quotient Rule:** $b'(x) = \dfrac{-\sin(x)\left(2 - x^9\right) - \cos(x)\left(-9x^8\right)}{\left(2 - x^9\right)^{-2}} = \dfrac{-\mathbf{sin}(x)}{2 - x^9} + \dfrac{9x^8 \mathbf{cos}(x)}{\left(2 - x^9\right)^2}$
   - $c(x) = \dfrac{\sin\left(3 + 4^x\right)}{2 - x^9} = \sin\left(3 + 4^x\right) \cdot \left(2 - x^9\right)^{-1}$
     - **Product Rule:**
     - **Quotient Rule:**
5. Using Quotient Rule and the Basic Derivative formulas for $\sin(x)$ and $\cos(x)$:
   - $\tan(x) = \dfrac{\sin(x)}{\cos(x)}$

$$
\begin{aligned}
[\tan(x)]' &= \frac{\cos(x)\cos(x) - (-\sin(x))\cos(x)}{\cos^2(x)} \\
&= \frac{\cos^2(x) + \sin^2(x)}{\cos^2(x)} \\
&= \frac{1}{\cos^2(x)} = \mathbf{sec}(x)
\end{aligned}
$$

- $\cot(x) = \dfrac{\cos(x)}{\sin(x)}$

$$\begin{aligned}
[\cot(x)]' &= \frac{(-\sin(x))\sin(x) - \cos(x)\cos(x)}{\sin^2(x)} \\
&= \frac{(-\sin(x))\sin(x) - \cos(x)\cos(x)}{\sin^2(x)} \\
&= \frac{-\left[\sin^2(x) + \cos^2(x)\right]}{\sin^2(x)} \\
&= \frac{-1}{\sin^2(x)} = -\mathbf{csc}(x)
\end{aligned}$$

- $\sec(x) = \dfrac{1}{\cos(x)}$

$$\begin{aligned}
[\sec(x)]' &= \frac{0 \cdot \cos(x) - 1 \cdot (-\sin(x))}{\cos^2(x)} \\
&= \frac{\sin(x)}{\cos^2(x)} \\
&= \frac{1}{\cos(x)} \cdot \frac{\sin(x)}{\cos(x)} = \mathbf{sec}(x)\,\mathbf{tan}(x)
\end{aligned}$$

- $\csc(x) = \dfrac{1}{\sin(x)}$

$$\begin{aligned}
[\csc(x)]' &= \frac{0 \cdot \sin(x) - 1 \cdot \cos(x)}{\sin^2(x)} \\
&= \frac{-\cos(x)}{\sin^2(x)} \\
&= \frac{-1}{\sin(x)} \cdot \frac{\cos(x)}{\sin(x)} = -\mathbf{csc}(x)\,\mathbf{cot}(x)
\end{aligned}$$

6. Find the interval(s) when the following functions are increasing, decreasing, concave up, and concave down.
   - $a(x) = 2x^3 + 3x^2 - 12x + 4$ on $[-4, 3]$
     - **Increasing:** $(-4, -2) \cup (1, 3)$
     - **Decreasing:** $(-2, 1)$
     - **Concave up:** $(-0.504, 3)$
     - **Concave down:** $(-4, -0.504)$
   - $b(x) = 3x + \sin(4x)$ on $[0, 1]$
     - **Increasing:** $(0, 0.605) \cup (0.966, 1)$
     - **Decreasing:** $(0.605, 0.966)$
     - **Concave up:** $(0.785, 1)$
     - **Concave down:** $(0, 0.785)$
7. Find the absolute and relative extrema of the following functions:
   - $a(x) = 2x^3 + 3x^2 - 12x + 4$ on $[-4, 3]$

  - **Relative max:** $(-2, 24)$
  - **Relative min:** $(1, -3)$
  - **Absolute max:** $(3, 49)$
  - **Absolute min:** $(-4, -28)$
- $b(x) = 3x + \sin(4x)$ on $[0, 1]$
  - **Relative max:** $(0.605, 2.476)$
  - **Relative min:** $(0.966, 2.237)$
  - **Absolute max:** $(0.605, 2.476)$
  - **Absolute min:** $(0, 0)$

8. Use the MVT to do the following problems:
   - The number(s) c which satisfy the conclusion of the MVT for the function $f(x) = 4x^3 - 8x^2 + 7x - 2$ on $[-3, 5]$ are $c = -\frac{5}{3}$ and $c = 3$.
   - Given $f(x)$ a continuous and differentiable function on $[-3, 2]$. It is known that $f(-3) = 5$ and $f'(x) \geq 7$. What is the smallest possible value for $f(2)$?
     - From the conclusion of MVT, we have $f(2) - f(-3) = f'(c)(2-(-3))$, so $f(2) = f(-3) + f'(c)(2+3) = 5 + 5f'(c)$.
     - Since $f'(x) \geq 7$ then $f'(c) \geq 7$. Thus $f(2) = 5 + 5f'(c) \geq 5 + 8(7) = 61$. Thus, the smallest possible value for $f(2)$ is 61.
9. Find the linearization for each given function at each given points, and plot the graph together with their linearization. For each, also use the linearization to approximate the value at the given nearby point, then calculate the error of the approximation.

   For the plot, you can use the given program in the module (Section 4.6).
   - $f(x) = \sqrt[4]{x}$ at $x = 16$, use to approximate $\sqrt[4]{16.7}$
     - **Linearization:** $L(x) = 2 + \dfrac{1}{32}(x - 16)$
     - **Approximate:** $L(16.7) \approx 2.0021875$
     - **Actual value:** $\sqrt[4]{16.7} \approx 2.021525$
     - **Approximate Error:** 0.019338
   - $g(x) = \sin(x)$ at $x = \dfrac{\pi}{4}$, use to approximate $\sin\left(\dfrac{3\pi}{11}\right)$
     - **Linearization:** $L(x) = \dfrac{\sqrt{2}}{2}\left(1 + \dfrac{\pi}{44}\right)$
     - **Approximate:** $L\left(\dfrac{3\pi}{11}\right) \approx 0.757594$
     - **Actual value:** $\sin\left(\dfrac{3\pi}{11}\right) \approx 0.755750$
     - **Approximate Error:** 0.001844
10. Use implicit differentiation to find y' for each of the following:
    - $5x + x^4y^3 = 7y^3 - 2 \Rightarrow \dfrac{dy}{dx} = \dfrac{5 + 4x^3y^3}{7y^2 - x^4 3y^2}$

- $3x^2 \tan(y) = x - y^5 \sec x \Rightarrow \dfrac{dy}{dx} = \dfrac{1 - y^5 \tan(x) \sec(x) - 6x \tan(y)}{3x^2 \sec^2(y) + 5y^4 \sec(x)}$
- $e^{x+2y} = \ln(xy^2) + x^2 \Rightarrow \dfrac{dy}{dx} = \dfrac{\frac{1}{x} + 2x - e^{x+2y}}{2e^{x+2y} - \frac{2}{y}}$

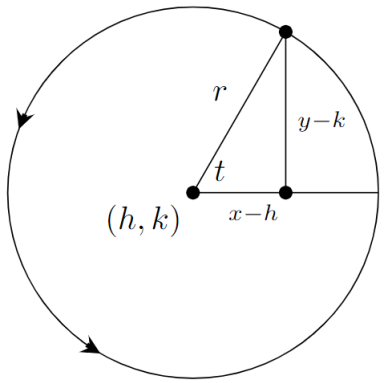
8.8.4 Integrals

1. Find the indefinite integral of the following functions.
   - $\displaystyle\int \sin(\cos(x)) \sin(x)\, dx = \cos(\cos(x)) + C$
   - $\displaystyle\int 3x \sqrt[3]{1 - 2x^2}\, dx = -\frac{9}{16}\left(1 - 2x^2\right)^{4/3} + C$
   - $\displaystyle\int \frac{3x^5 + 2x}{\left(x^6 + 2x^2\right)^3}\, dx = -\frac{1}{4}\left(x^6 + 2x^2\right)^{-2} + C$
   - $\displaystyle\int x^5 \sqrt{x^3 + 1}\, dx = \frac{2}{45}\left(x^3 + 1\right)^{3/2}\left(3x^2 - 2\right) + C$
   - $\displaystyle\int x^2 \cos(7x)\, dx = \frac{1}{343}\left(\left(49x^2 - 2\right)\sin(7x) + 14x\cos(7x)\right) + C$
2. For each of the following function at the given interval, find the Riemann Sum with $n = 8$, using Left, Midpoint, and Right endpoints for each.
   - $f(x) = \sin(x)$ on $[0, \pi]$
     - **Left Sum** ≈ 1.9742316
     - **Midpoint Sum** ≈ 2.01290909
     - **Right Sum** ≈ 1.9742316
   - $g(x) = 3x^3 + 5x - 7$ on $[-2, 6]$
     - **Left Sum** $= 652$
     - **Midpoint Sum** $= 972$
     - **Right Sum** $= 1364$
   - $h(x) = x^5\sqrt{x^3 + 1}$ on $[2, 10]$
     - **Left Sum** ≈ 2808581.720
     - **Midpoint Sum** ≈ 4134814.161
     - **Right Sum** ≈ 5972344.124
3. Use FTC 2 to compute the following, and determine which Sum (Left/Midpoint/Right) on the previous problem gives the best approximation.
   - $\displaystyle\int_0^{\pi} \sin(x)\, dx = 2$ (**Midpoint** Sum gives best approximation)
   - $\displaystyle\int_{-2}^{6} 3x^3 + 5x - 7\, dx = 984$ (**Midpoint** Sum gives best approximation)
   - $\displaystyle\int_2^{10} x^5\sqrt{x^3 + 1}\, dx \approx 4219854.819$ (**Midpoint** Sum gives best approximation)

4. Use FTC 1 to differentiate each of the following integral with respect to x.
   - $\displaystyle\int_2^x 7\sin^2\left(t^3-0.2t+9\right)\,dt = 7\,\mathbf{sin}^2\left(x^3-0.2x+9\right)$
   - $\displaystyle\int_2^{\cos(3x)} \sqrt{17-9t^6}\,dt = -3\,\mathbf{sin}(3x)\sqrt{17-9\,\mathbf{cos}^6(3x)}$
   - $\displaystyle\int_{-5x}^{5x^2} e^t-\ln(4t)\,dt = 5\left(e^{-5x}-\mathbf{ln}(-20x)\right)+10x\left(e^{5x^2}-\mathbf{ln}\left(20x^2\right)\right)$
5. Determine if the following integral is convergent or divergent. If convergent, calculate its value. If divergent, explain why.
   - $\displaystyle\int_4^\infty \frac{2}{x^3}\,dx = \frac{1}{16}$ (is **convergent**)
   - $\displaystyle\int_{-\infty}^5 e^{2x}+1\,dx$ is **divergent**
   - $\displaystyle\int_{-\infty}^\infty \frac{x^2}{x^3+1}\,dx$ is **divergent**
6. Find the average value of the given function on the given interval.
   - $f(x)=10\ln(x)+x^2$ on $[3,5]$
     - $\displaystyle\frac{1}{5-3}\int_3^5 10\ln(x)+x^2\,dx \approx 30.09$
   - $t(x)=70+950e^{-0.07x}$ on $[0,60]$
     - $\displaystyle\frac{1}{60-0}\int_0^{60} 70+950e^{-0.07x}2\,dx \approx 292.799$
   - $g(x)=-\sin\left(\frac{1}{2}x\right)+\cos(2x)$ on $\left[\frac{\pi}{2},\pi\right]$
     - $\displaystyle\frac{1}{\pi-\frac{\pi}{2}}\int_{\pi/2}^{\pi} -\sin\left(\frac{1}{2}x\right)+\cos(2x)\,dx \approx -0.203231$
7. The number c that satisfies the MVT for Integrals, for $f(x)=4x^2+5x+6$ on $[-1,5]$ is $c\approx 2.52$.
8. Find the area of the following:
   - Area between $f(x)=3x$ and $g(x)=x\left(\sqrt{x}+1\right)$ for $0\le x\le 4$ is $\dfrac{4\left(89-25\sqrt{5}\right)}{15}\approx 8.82621$
   - Area between $f(y)=3y$ and $g(y)=30y\cdot e^{-0.3y}$ for $0\le y\le 9$ is ≈ 140.818.
   - Area enclosed by $f(x)=4-x$ and $g(x)=(x-1)^2$ is $\dfrac{13\sqrt{13}}{6}\approx 7.81203$.

8.8.5 Parametric Equations

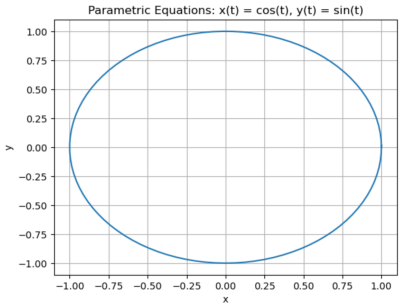
1. Show that for a circle centered at $(h, k) \neq (0, 0)$, the parametric equation is $x = h + r\cos(t)$ and $y = k + r\sin(t)$.



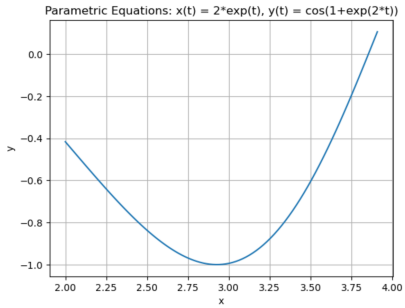
In the right triangle: $\sin(t) = \dfrac{y-k}{r}$ and $\cos(t) = \dfrac{x-h}{r}$. The rest follows by simple algebraic manipulation.

2. Sketch the following parametric equations. What shape is the curve?
   Can use the code on Section 6.1 to plot the curve.

   - $x = \cos(t)$ and $y = \sin(t)$ for $0 \leq t \leq 2\pi$. The curve shape is **(unit) circle**.

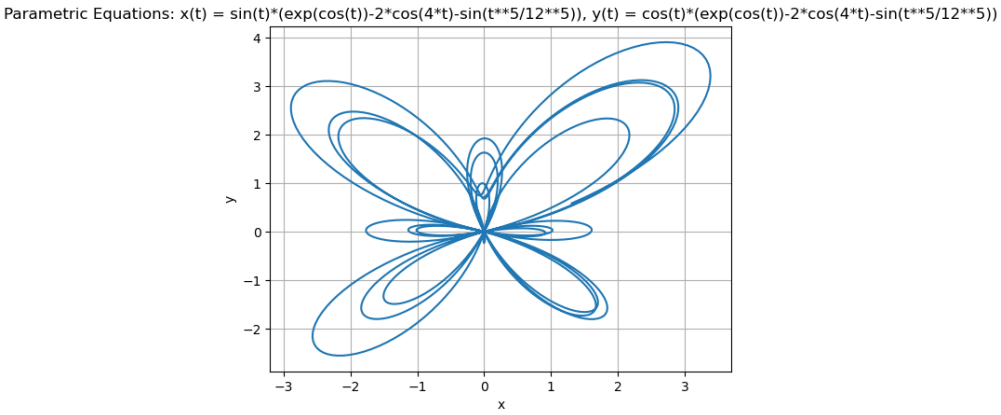


   - $x = 2e^t$ and $y = cos\left(1 + e^{2t}\right)$ for $0 \leq t \leq \dfrac{2}{3}$. The curve is **open up parabola-ish**.



   - $x = \sin(t) \cdot \left(e^{\cos(t)} - 2\cos(4t) - \sin^5\left(\dfrac{t}{12}\right)\right)$ and $y = \cos(t) \cdot \left(e^{\cos(t)} - 2\cos(4t) - \sin^5\left(\dfrac{t}{12}\right)\right)$ for $0 \leq t \leq 20$.

     This is known as the **Butterfly curve**.

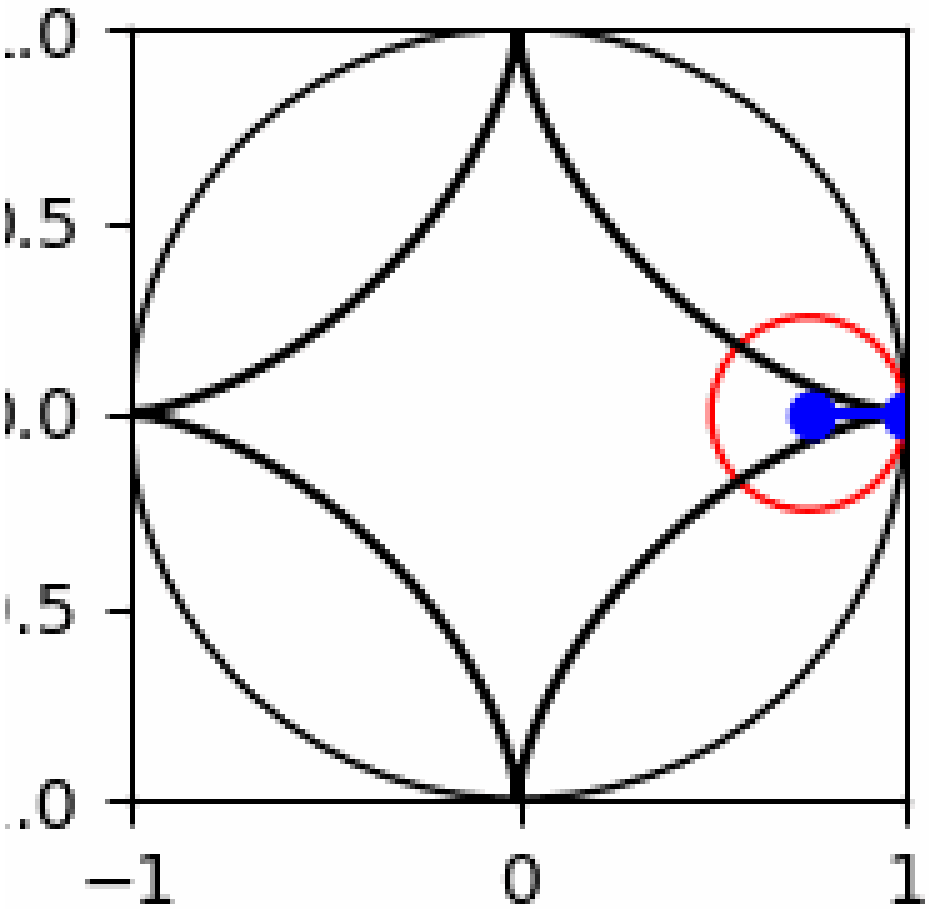


3. Given the parametric equations $x(t) = t^2 - 2t - 11$ and $y(t) = -2t^4 + 12t^3 - 64t + 7$.
   - The tangent line(s) at the point $(-3, 7)$ are $y = -24x - 65$ and $y = 7$.
   - The (x, y)-coordinates of the three horizontal tangent are $(-3, 7)$, $\left(\frac{-75 + 3\sqrt{33}}{8}, \frac{-5 + 165\sqrt{33}}{16}\right)$, $\left(\frac{-75 - 3\sqrt{33}}{8}, \frac{-5 -$

     The (x, y)-coordinates of the only vertical tangent is $(-12, -47)$.
   - $\frac{d^2y}{dx^2} = \frac{-8t^3 + 30t^2 - 36t + 32}{(t-1)^3}$.
4. Given the parametric equations $x(t) = 3 - \cos^3(t)$ and $y(t) = 4 + \sin(t)$.
   - The area below the parametric curve on $0 \le t \le \pi$ is $A = 8 + \frac{3\pi}{8}$.
   - The arc length of the parametric curve on $0 \le t \le 4\pi$ is the arc length of the parametric curve on $0 \le t \le 2\pi$ (self-overlapping), which is $L \approx 5.82896$.
5. Do the following problems:
   - $(r, \theta) = \left(-2, \frac{2\pi}{3}\right)$ in Cartesian coordinates is $(x, y) = \left(1, -\sqrt{3}\right)$.
   - $(x, y) = (-2\sqrt{3}, -6)$ in Polar coordinates is $(r, \theta) = \left(\sqrt{48}, \frac{4\pi}{3}\right)$.
   - $6r^3 \sin(\theta) + \cos(\theta) = 4$ in Cartesian coordinates is $6y\left(x^2 + y^2\right)^{3/2} + x = 4\sqrt{x^2 + y^2}$.
   - $6 - xy = \frac{2x}{5x^2 + 5y^2}$ in Polar coordinates is $6 - r^2 \cos(\theta) \sin(\theta) = \frac{2r \cos(\theta)}{5r^2}$.
6. The equation of the tangent line to $r = \cos(\theta)\sin(4\theta)$ at $\theta = \frac{\pi}{6}$ is $y - \frac{3}{8} = \frac{\sqrt{3}}{9}\left(x - \frac{3\sqrt{3}}{8}\right)$, or, simplified to $y = \frac{\sqrt{3}}{9}x + \frac{1}{4}$.
7. The area that is inside both $r = 2 + \sin(\theta)$ and $r = 1 - \sin(\theta)$ is $\frac{5\pi}{2} - 3\sqrt{3} \approx 2.6578$.

   Using the Hint: Area1 is area inside $r = 2 + \sin(\theta)$, while Area2 is the area outside $r = 2 + \sin(\theta)$ but inside $r = 1 - \sin(\theta)$.
8. Determine the arc length of the curve for each given polar equation on each given θ range.
   - $r = \theta$ for $0 \le \theta \le 2\pi \Rightarrow L \approx 21.256$.
   - $r = \cos(\theta) + \sin(3\theta)$ for $0 \le \theta \le \frac{2\pi}{3} \Rightarrow L \approx 5.287$.

JNB Supplement



Note

Please visit the web version of this section to engage with this interactive resource.

8.8.6 Sequences and Series

1. List the first 5 terms of each given sequence, then determine whether each sequence is monotonic and/or bounded.
   - $\left\{\frac{n}{n+1}\right\}_{n=4}^{\infty} = \frac{4}{5}, \frac{5}{6}, \frac{6}{7}, \frac{7}{8}, \frac{8}{9}, \ldots$ is **monotonic (decreasing)** and **bounded**
   - $\left\{\frac{2n^2-1}{n}\right\}_{n=2}^{\infty} = \frac{7}{2}, \frac{17}{3}, \frac{31}{4}, \frac{49}{5}, \frac{71}{6}, \ldots$ is **monotonic (increasing)** and **not bounded** (no upper bound).
2. Determine whether the following sequences converge or diverge. Determine the limit if it exists.
   - $\left\{\frac{(-1)^n \cdot n^3}{5-n^4}\right\}_{n=0}^{\infty}$ **converges**, $\lim_{n\to\infty} \frac{(-1)^n \cdot n^3}{5-n^4} = 0.$

- $\left\{\dfrac{e^{3n}}{2+e^n}\right\}_{n=1}^{\infty}$ **diverges**, to ∞.
- $\left\{\dfrac{\ln(n+6)}{\ln(3+3n)}\right\}_{n=1}^{\infty}$ **converges**, $\lim\limits_{n\to\infty}\dfrac{\ln(n+6)}{\ln(3+3n)}=1$.

3. Index shift and rename exercise:
   - $\sum\limits_{i=1}^{\infty}\dfrac{5-i}{i^3}=\sum\limits_{n=3}^{\infty}\dfrac{5-(n-2)}{(n-2)^3}=\sum\limits_{n=3}^{\infty}\dfrac{7-n}{(n-2)^3}$.
   - $\sum\limits_{i=5}^{\infty}(-1)^{i+2}2^i=\sum\limits_{k=1}^{\infty}(-1)^{(k+4)+2}2^{k+4}=\sum\limits_{k=1}^{\infty}(-1)^{k+6}2^{k+4}$.
4. Determine the convergence of the following series .
   - $\sum\limits_{i=1}^{\infty}\dfrac{1}{i^{\pi/4}}$ is **divergent**
   - $\sum\limits_{i=1}^{\infty}\dfrac{i-1}{\sqrt{i^4+2}}$ is **convergent**
   - $\sum\limits_{i=1}^{\infty}\dfrac{(-1)^{i+2}}{i^3+2i+2}$ is **convergent**
5. Determine the interval and radius of convergence of the following series:
   - $\sum\limits_{i=1}^{\infty}\dfrac{4^{2i+1}}{5^{1+i}}(x+3)^i$ **Radius** $R=\dfrac{5}{16}$, **Interval** $-\dfrac{53}{16}<x<-\dfrac{43}{16}$.
   - $\sum\limits_{i=1}^{\infty}\dfrac{(i+1)}{(2i+1)!}(x-2)^i$ **Radius** $R=\infty$, **Interval** $-\infty<x<\infty$.

Note

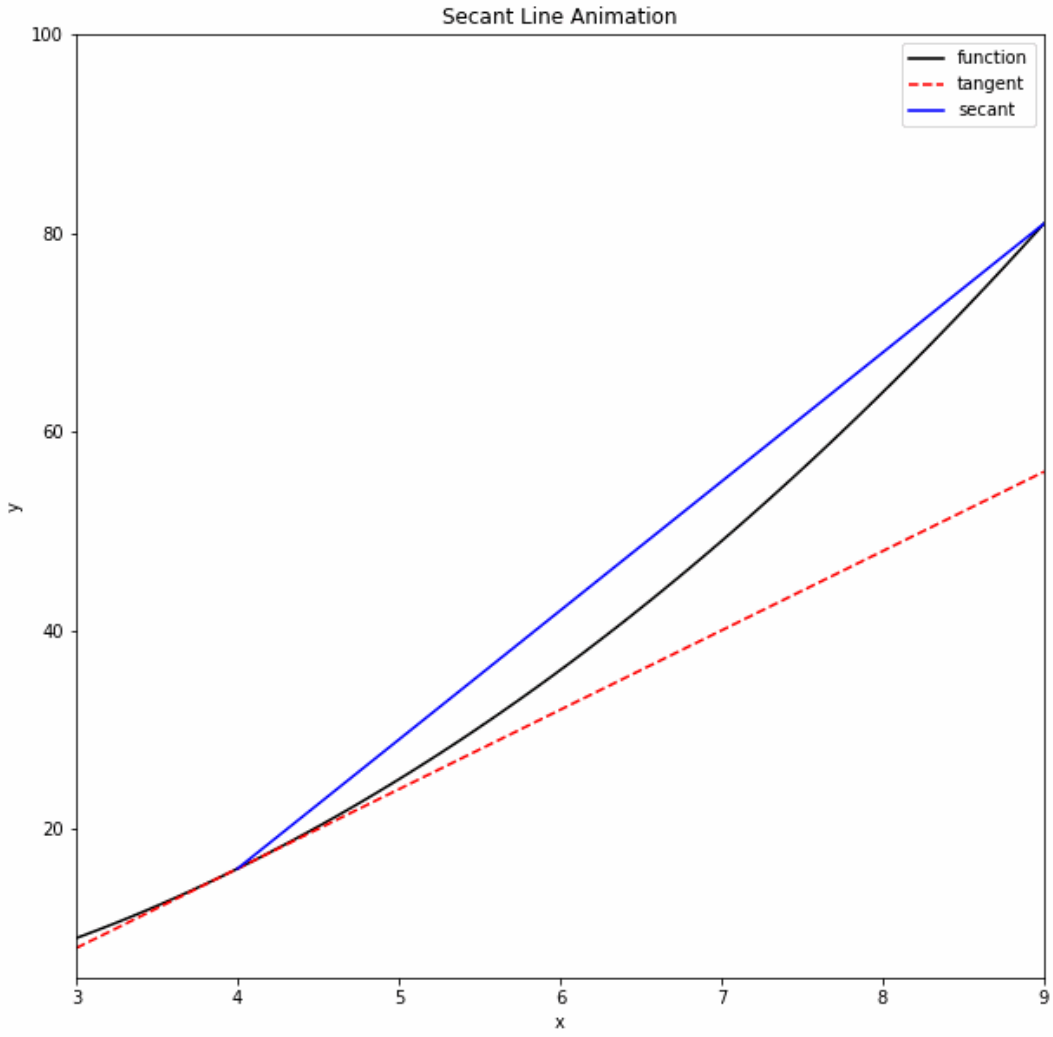
Please visit the web version of this section to engage with this interactive resource.

8.9 JNB LAB: Calculus Animations

In this JNB, we will show how to create animations that illustrate basic calculus concepts. To create an animation, we use a **for** loop to create single frames of the animation '0.png', '1.png', '2.png',.... and then use a block of code to combine the individual frames into a .gif animation.

8.9.1 Tangent Lines

Let's begin by showing how secants approach a tangent line. We will use the function $f(x)=x^2$ and create a sequence of secants that approach the tangent line at point $x=4$.



First, let's define our function and specify the point of tangency.

```
f = lambda x : x**2 # Define the function f(x)
a=4   #x coordinate of point of tangency
```

Now let's create the first frame ($n = 0$) of the sequence of images $\{n.png\}$ used to create the animation.

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

To create our animation, we will:

i) import a couple of functions used to create the animation

ii) specify the number of frames

iii) specify our function and point of tangency (assuming we haven't done so previously)

iv) use a for loop to create the individual frames in the animation

v) combine the frames into an animation file "secant.gif" which is found in the same folder as this lab.

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

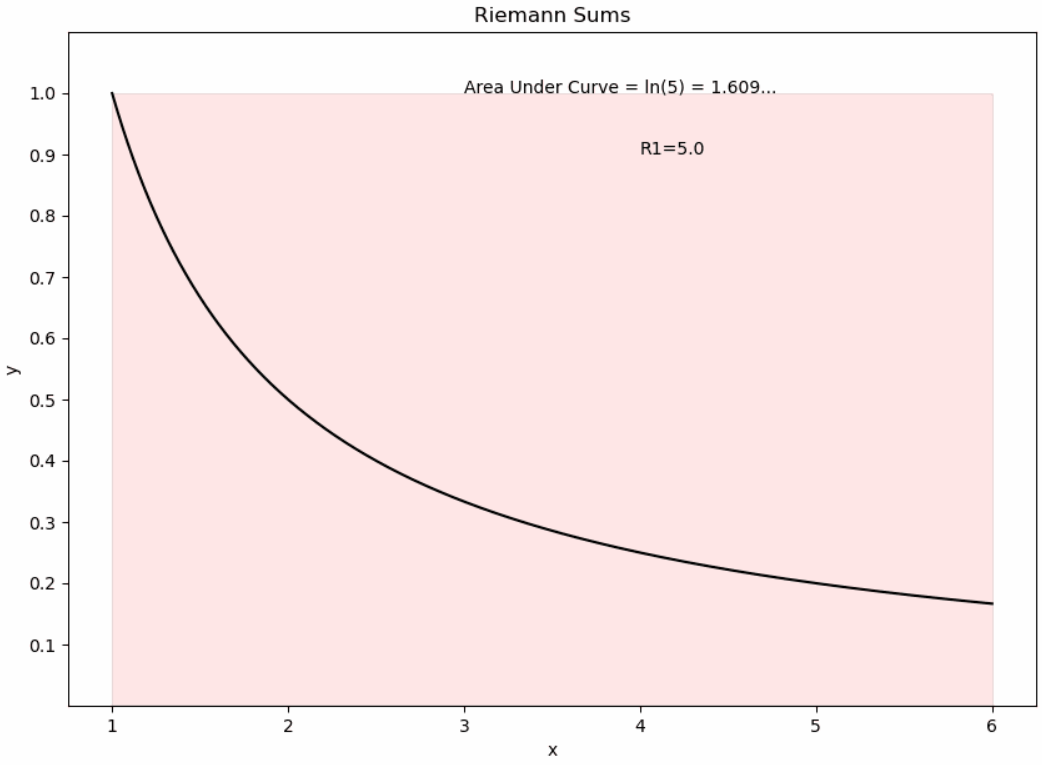
Exercise

Exercise

1.1 Create an animation with 3 frames which shows secants approaching the tangent line to $y = \frac{1}{x}$ at $x = 1$.

8.9.2 Riemann Sums

Just as the tangent line is obtained by a limiting process, the area under curve is obtainable as a limit of a Riemann Sum. We can create an animation which shows the approximating rectangles filling in the area under a curve. For example, we consider the area under $y = 1/x$ in the interval $1 \leq x \leq 5$.




```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from matplotlib.animation import FuncAnimation
%matplotlib notebook
f = lambda x : 1/x
x=np.arange(1,6,.0001)
y=f(x)
frames=10
for n in np.arange(0,frames,1):
    plt.figure(figsize=(10,7))
    plt.gca().set_xticks(np.arange(1,7,1))
    plt.gca().set_yticks([.1,.2,.3,.4,.5,.6,.7,.8,.9,1])
    plt.plot(x,y,'k',markersize=1)
    x_left = np.linspace(1,6,n+2)
    x_left=x_left[:-1]
    plt.bar(x_left,1/x_left,width=5/(n+1),alpha=1-.9*(1/(n+1)),align='edge',color='r',
 ↪edgecolor='k')
    plt.title('Riemann Sums')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.ylim((0,1.1))
    plt.text(3,1,"Area Under Curve = ln(5) = 1.609...")
    Rn=0
    for k in np.arange(0,n+1,1):
        Rn=Rn+(5/(n+1))*(1/x_left[k])
```

(continues on next page)

(continued from previous page)

```
    plt.text(4,.9, "R"+str(n+1)+"="+str(Rn))
    #plt.grid()
    plt.savefig(str(n)+'.png')
    plt.show()
#----- Save the Animation as riemannsums.gif
from PIL import Image
images = []
for n in range(frames):
    exec('a'+str(n)+'=Image.open("'+str(n)+'.png")')
    images.append(eval('a'+str(n)))
images[0].save('riemannsums.gif',
               save_all=True,
               append_images=images[1:],
               duration=800,
               loop=0)
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

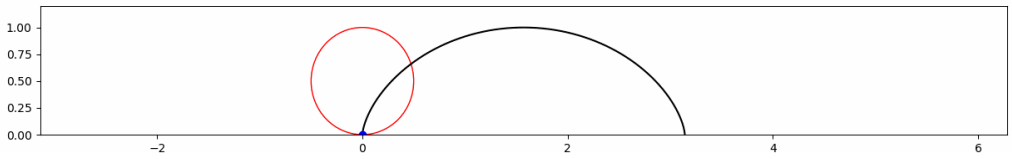
Exercise

Exercise

2.1 Create an animation for a Riemann sum approximation of $\int_0^{\pi/2} \cos(x)\,dx$.

8.9.3 Parametric Curves

JNBs can be used to create .gif animations of parametric curves. For example, the code below is used to animate a cycloid curve traced by a point on the rim of a wheel.



Exercise

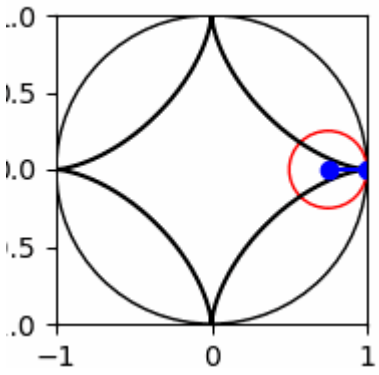
Exercise

3.1 A hypocycloid curve is formed by a point on a circle rotating along the inside rim of another circle. The parametric equations are

$x(t) = (a-b)\cos t + b\cos(\frac{a-b}{b}t)$

$y(t) = (a-b)\sin t - b\sin(\frac{a-b}{b}t)$

Create an animation of a hypocycloid curve for the case $a = 1$, $b = 1/4$. Your animation should resemble the one shown below.

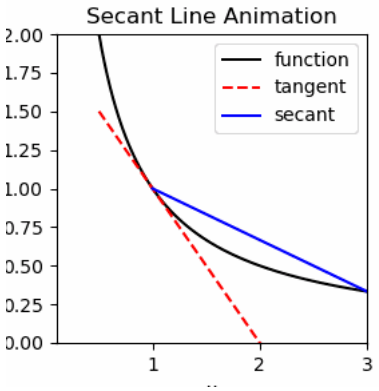


Note

Please visit the web version of this section to engage with this interactive resource.

8.10 JNB Lab Solutions

Exercise 1.1




```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

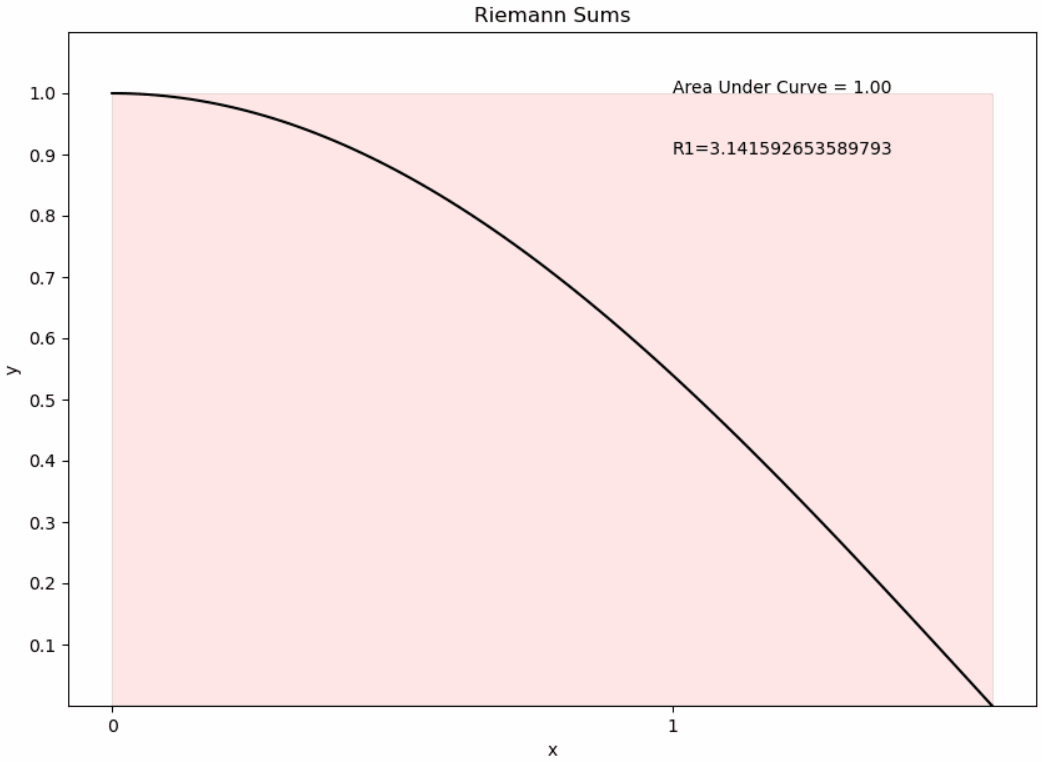
```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

2.1




```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

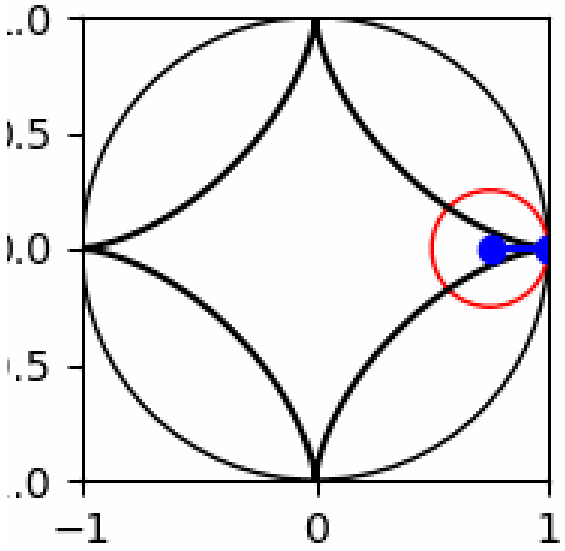
```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

3.1




```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
C:\Users\pisihara\AppData\Local\Temp\ipykernel_19208\169837878.py:7:␣
 ↪RuntimeWarning: More than 20 figures have been opened. Figures created through␣
 ↪the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly␣
 ↪closed and may consume too much memory. (To control this warning, see the␣
 ↪rcParam `figure.max_open_warning`). Consider using `matplotlib.pyplot.close()`.
  fig = plt.figure(figsize=(2, 2))
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

CHAPTER 9

LINEAR ALGEBRA - Soheil Anbouhi

Note

Dr. Anbouhi covers the fundamentals of linear algebra. The JNB lab was contributed by Gavin McClowry.

9.1 Introduction

This chapter serves as an introductory exploration of fundamental concepts in linear algebra with a focus on computational aspects. It also serves as the foundational background for the subsequent chapter, "Linear Algebra and Optimization for Data Analysis".

Each section begins with a brief overview of the objectives and is followed by several example exercises to enhance understanding and facilitate learning. Moreover, we strive to include numerical notes wherever appropriate. These notes aim to explain and compare different computational approaches to problem-solving. The goal is for users to write their own code (whenever possible) to solve problems. However, for matrices of higher dimensions, we take advantage of the NumPy linear algebra module which provides efficient implementations of standard linear algebra algorithms.

The chapter covers the following topics:

1. Linear Systems
2. Matrices and Determinants
3. Linear Transformations
4. Eigenvalues and Eigenvectors
5. Orthogonality

We will begin by exploring systems of linear equations, also known as **linear systems**, which play a central role in linear algebra. In fact, many problems in linear algebra can be converted into solving a linear system.

Linear systems and their solutions can be described using **matrices**. We will study the algebraic properties of matrices in Section 2. These properties can be used to solve a linear system and set a groundwork for topics covered in the following sections.

In Section 3, we will discuss **linear transformation** between subspaces of $\mathbb{R}^n$, and explore how changes of bases affect the representation of these transformations. We won't delve into vector spaces in their general form. This approach maintains generality as an n-dimensional vector space and $\mathbb{R}^n$ are the same from a computational point of view.

Section 4 is devoted to dissecting the action of linear maps into elements that are easy to visualize, using **eigenvalues** and **eigenvectors**. We will delve into how a basis formed by eigenvectors leads to diagonal matrices (diagonalization). We will also examine the application of diagonalization in dealing with large matrix products and discrete dynamical systems.

Section 5 explores some ideas from analytic geometry, which allows us to define intuitive geometric concepts such as length, distance, and perpendicularity in $\mathbb{R}^n$. It will also discuss orthogonal projection, the Gram-Schmidt algorithm, and the QR- factorization which are key concepts in many calculations involving **orthogonality**. Finally, we will discuss the regression analysis concept of the least squares problem to approximate the solution of linear systems in which there are more equations than unknowns.

The content of this chapter is based on:

- Lay, David, et al. Linear Algebra and its Applications. 5th ed., Pearson., 2016
- Deisenroth, Marc Peter, A. Aldo Faisal, and Cheng Soon Ong. Mathematics for Machine Learning. Cambridge University Press, 2020.

```
import matplotlib.pyplot as plt
import numpy as np
```

9.2 Linear Systems

9.2.1 Systems of Linear Equations

This section introduces systems of linear equations, or linear systems, and discusses how to find their solutions.

Linear Systems

A *linear equation* in n variables is an equation of the form:

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = b$$

where $x_1, x_2, \dots, x_n$ are variables, and b and the coefficients $a_1, a_2, \dots, a_n$ are real numbers. A *system of linear equations* is a collection of linear equations in the same set of variables. For example, a system of m equations in n variables $x_1, x_2, \dots, x_n$ can be represented as:

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \tag{9.1}$$
$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \tag{9.2}$$
$$\vdots \tag{9.3}$$
$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \tag{9.4}$$
$$\tag{9.5}$$

A *solution* for a linear system is a list of real numbers $(s_1, s_2, \dots, s_n)$ that satisfy each equation in the system. Two linear systems are said to be equivalent if they have the same set of solutions. In other words, every solution of the first system is also a solution of the second system, and vice versa. Furthermore, a linear system is called **consistent** if it has solutions; otherwise, it is called **inconsistent**. It turns out that any linear system has either no solutions, exactly one solution, or infinitely many solutions.

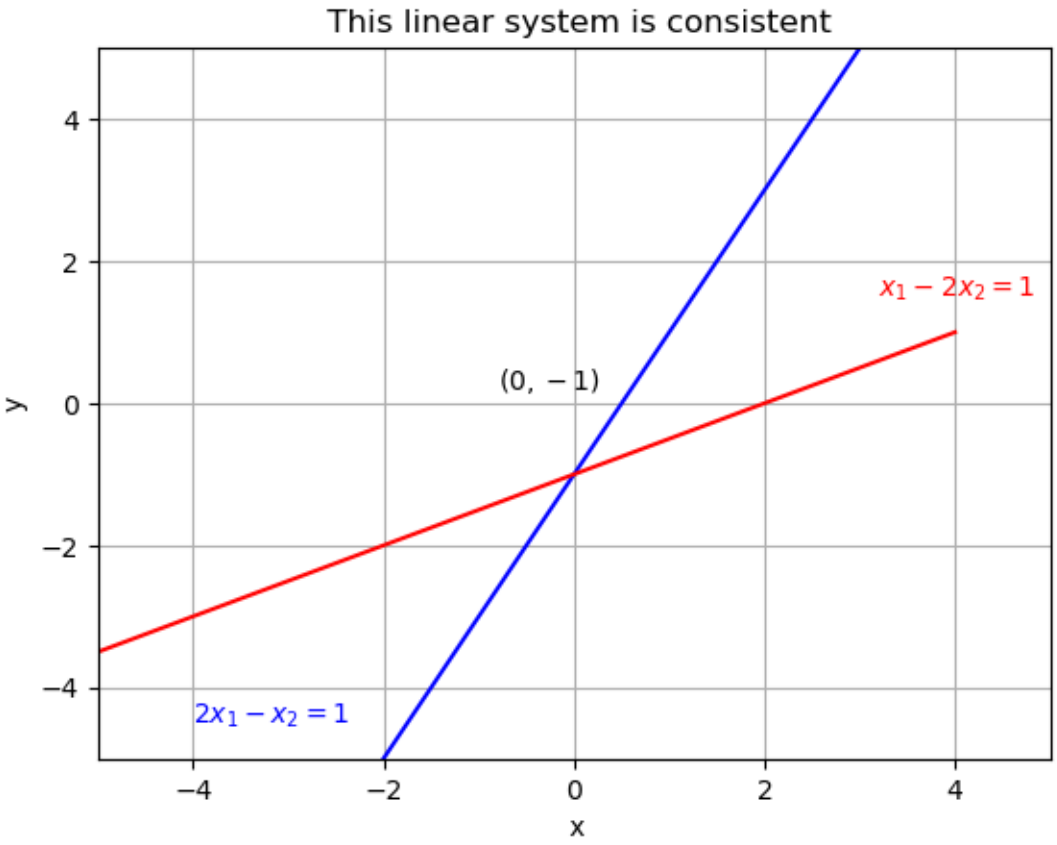
As an example, let's consider a simple case where we have two equations in two unknowns. Finding the solution of such a system is easy as it comes down to finding the intersection of two lines in the plane. More precisely, the graph of each equation in this system represents a line, and a point $(s_1, s_2) \in \mathbb{R}^2$ is a solution if and only if it lies on both lines.

Example 1

The following system is consistent, and its only solution is $(0, -1)$.

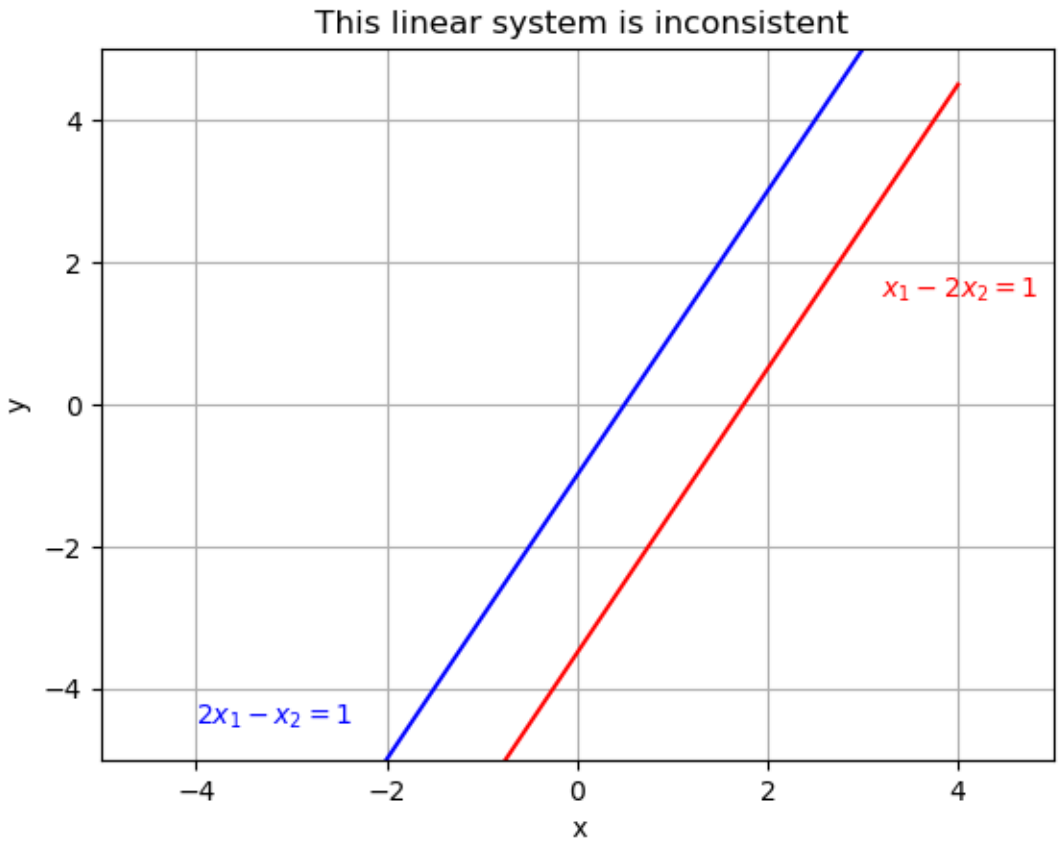
$$2x_1 - x_2 = 1$$
$$x_1 - 2x_2 = 2$$

The following cell plots the graph of these lines on the same plane:



(2) The next system is inconsistent because it does not have any solutions.

$$2x_1 - x_2 = 1$$
$$4x_1 - 2x_2 = 7$$



Similarly, we can find a linear system containing two equations in two unknowns that has infinitely many solutions (see Exercise 1).

9.2.2 Solving a Linear System

A fundamental question about a linear system is whether or not it is consistent and, if it is, whether it has a unique solution or infinitely many solutions. To that end, we utilize the **row reduction algorithm**. The main idea is to use a set of operations called *elementary row operations* to convert a system into an equivalent system that is easier to solve. These elementary row operations are:

1. Swapping two equations
2. Multiplying an equation by a nonzero number
3. Adding a multiple of one equation to another equation

To do this, we first represent the essential information of a system in a compact rectangular form called the augmented matrix of the system. For example, the augmented matrix of the system:

$$\begin{aligned} x + 2y + 3z &= 9 \\ 2x - y + z &= 8 \\ 3x - z &= 3 \end{aligned}$$

is the following matrix:

$$\begin{bmatrix} 1 & 2 & 3 & 9 \\ 2 & -1 & 1 & 8 \\ 3 & 0 & -1 & 3 \end{bmatrix}$$

In the augmented matrix, each row represents an equation in the system. We can define the same operations for matrices, by simply replacing "equation" with "row" in their definitions. Moreover, we say that two matrices A and B are **row equivalent** if one can be transformed into the other using the elementary row operations mentioned above.

The goal is to convert the augmented matrix to another matrix form that is easier to solve. This easier form is called **echelon form**. This new matrix form corresponds to a linear system that is simpler to deal with and is equivalent to the original system.

To formally define the echelon form, we need to introduce the following concepts:

Zero Row: a row containing only zero entries.

Leading Entry: the leftmost nonzero entry in a nonzero row.

A rectangular matrix is in row echelon form (REF) if:

1. All non-zero rows are above any zero row
2. Every leading term of a row is in a column to the right of the leading entry
3. Entries below a leading entry are zero

A rectangular matrix is in **reduced row echelon form** (RREF) if:

1. It is in echelon form
2. Every leading entry is 1
3. Entries above a leading entry are zero

Example 2

The following matrix is in REF. The leading entries (■) can be any nonzero real numbers, while the (∗s) may be any real number.

$$\begin{bmatrix} 0 & \blacksquare & * & * & * & * & * \\ 0 & 0 & \blacksquare & * & * & * & * \\ 0 & 0 & 0 & 0 & \blacksquare & * & * \\ 0 & 0 & 0 & 0 & 0 & \blacksquare & * \\ 0 & 0 & 0 & 0 & 0 & 0 & \blacksquare \end{bmatrix}$$

The following matrix is in RREF:

$$\begin{bmatrix} 0 & 1 & 0 & * & 0 & * \\ 0 & 0 & 1 & * & 0 & * \\ 0 & 0 & 0 & 0 & 1 & * \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Theorem 1

Any nonzero matrix is row equivalent to one and only one matrix in RREF.

Reduction Algorithm:

The reduction algorithm takes in a matrix and produces a matrix in RREF. It consists of 5 steps; in step four, it produces a matrix in REF, and in the fifth step, RREF. We will show the algorithm with an example:

Example 3

Find the RREF of

$$A = \begin{bmatrix} 0 & 2 & 3 & 9 \\ 2 & -1 & 1 & 8 \\ 3 & 0 & -1 & 3 \end{bmatrix}$$

We represent a matrix as a numpy array

```
#numpy array to represent a matrix: each row is a list.
A = np.array([[0,2,3,9],[2,-1,1,8],[3,0,-1,3]])
A
```

```
array([[ 0,  2,  3,  9],
       [ 2, -1,  1,  8],
       [ 3,  0, -1,  3]])
```

Lets write Pythons functions that perform the row operations:

Step 1: Start with the leftmost nonzero entry in the first column and bring it to the top row. This is a pivot column; the pivot position should be on top.

```
array([0, 2, 3])
```

Step 2: Select a nonzero entry in the pivot column as a pivot. If necessary, swap the rows to move this entry to the top.

```
array([[ 2., -1.,  1.,  8.],
       [ 0.,  2.,  3.,  9.],
       [ 3.,  0., -1.,  3.]])
```

Step 3: Use scaling operation to make the pivot 1, and replacement operation to create zeros below the leading 1:

```
array([[ 1. , -0.5,  0.5,  4. ],
       [ 0. ,  2. ,  3. ,  9. ],
       [ 0. ,  1.5, -2.5, -9. ]])
```

Step 4: Ignore the row with a pivot position and cover all rows, if any, above it. For the remaining matrix, apply steps 1-3. Repeat the process until there are no more non-zero rows to modify.

```
array([[  1.  ,  -0.5 ,   0.5 ,   4.  ],
       [  0.  ,   1.  ,   1.5 ,   4.5 ],
       [  0.  ,   0.  ,  -4.75, -15.75]])
```

We apply the fifth step to get the RREF:

Step 5: If a pivot is not one, we use the scaling operation to make it one. Moreover, we create zeros above each pivot starting with the rightmost pivot and work upward and to the left.

```
array([[ 1.        ,  0.        ,  0.        ,  2.10526316],
       [ 0.        ,  1.        ,  0.        , -0.47368421],
       [-0.        , -0.        ,  1.        ,  3.31578947]])
```

A_9 is the RREF of A, and the augmented column (the rightmost column) of A_9 is the solution to the original system.

Example 4

Find the solution set of the following system of equations:

$$\begin{aligned} 2x - 5y + z &= 2 \\ x - y &= 4 \\ 3x - z &= 3 \end{aligned}$$

Solution: Let's set up the augmented matrix of the system and then convert it to its RREF:

```
array([[  1.,  -1.,   0.,   4.],
       [  0.,  -3.,   1.,  -6.],
       [  0.,   0.,   0., -15.]])
```

R4 is in REF. Now we write back the corresponding linear system:

$$\begin{aligned} x + y &= 4 \\ -3y + z &= -6 \\ 0 &= -15 \end{aligned}$$

The last equation is impossible, which means the system is inconsistent.

Exercises

Exercises

1. Find a linear system containing two equations in two unknowns that has infinitely many solutions. Verify your answer by plotting the lines of these equations on the same plane.
2. Find a linear system that is equivalent to the linear system in Example 1.
3. Determine which of the following matrices are in REF. For those that are, identify their pivot columns. Are any in RREF?

$$A = \begin{bmatrix} 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4. For the above matrices write the corresponding linear systems and find their solutions.
5. Find the REF of $D = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 3 \\ 4 & -3 & 8 \end{bmatrix}$.

9.2.3 Matrix Equation.

Note

Linear systems can be represented by equations involving vectors and matrices. In this section, we use such representations to study and analyze linear systems.

Let n be a positive integer, and $\mathbb{R}^n$ be the collection of all ordered n-tuples of real numbers. An element of $\mathbb{R}^n$, denoted by an $n \times 1$ **column matrix**, and is called a vector. For example,

$$\vec{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}$$

A vector whose entries are all zero is called **the zero vector** and is denoted by $\vec{0}$. When there is no risk of confusion we sometimes omit the arrow and use simply 0

Two vectors $\vec{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}$ and $\vec{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$ are **equal** if their corresponding entries are equal, that is if $u_i = v_i$ for all $i \leq n$.

Given a vector $\vec{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}$ and a real number c, the **scalar product** of $\vec{u}$ by c is given by:

$$c\vec{u} = \begin{bmatrix} cu_1 \\ cu_2 \\ \vdots \\ cu_n \end{bmatrix}$$

Theorem 2

Suppose $\vec{u}$, $\vec{v}$, $\vec{w}$ are in $\mathbb{R}^n$, and c and d are real numbers:

1. $\vec{u} + \vec{v} = \vec{v} + \vec{u}$
2. $(\vec{u} + \vec{v}) + \vec{w} = \vec{u} + (\vec{v} + \vec{w})$
3. $\vec{u} + \vec{0} = \vec{0} + \vec{u} = \vec{u}$
4. $\vec{u} + (-\vec{u}) = \vec{0}$
5. $c(\vec{u} + \vec{v}) = c\vec{u} + c\vec{v}$
6. $(c + d)\vec{u} = c\vec{u} + d\vec{u}$
7. $c(d\vec{u}) = (cd)\vec{u}$

Vector Equation

A vector $\vec{y} \in \mathbb{R}^n$ is called a **linear combination** of some vectors $\vec{u_1}, \vec{u_2}, \ldots, \vec{u_p} \in \mathbb{R}^n$, if there are real numbers $c_1, c_2, \ldots, c_p$ such that

$$\vec{y} = c_1\vec{u_1} + c_2\vec{u_2} + \cdots + c_p\vec{u_p}$$

The **span** of vectors $\vec{u_1}, \vec{u_2}, \ldots, \vec{u_p}$, denoted by $span\ (\vec{u_1}, \vec{u_2}, \ldots, \vec{u_p})$, is the set of all possible linear combinations:

$$span\ (\vec{u_1}, \vec{u_2}, \ldots, \vec{u_p}) = \{c_1\vec{u_1} + c_2\vec{u_2} + \cdots + c_p\vec{u_p} : \ c_1, c_2, \ldots, c_p \in \mathbb{R}\}$$

In other words, the span represents all the vectors that can be formed by scaling and adding the given vectors $\vec{u_1}, \vec{u_2}, \ldots, \vec{u_p}$.

Example 1

Let $\vec{y} = \begin{bmatrix} 7 \\ -4 \\ 3 \end{bmatrix}$, $\vec{u} = \begin{bmatrix} 2 \\ -3 \\ -5 \end{bmatrix}$, and $\vec{v} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$.

a. Compute $2\vec{u} - 3\vec{v}$. b. Determine if $\vec{y}$ is a linear combination of $\vec{u}$ and $\vec{v}$.

Solution:

(a).

$$2\vec{u} - 3\vec{v} = 2\begin{bmatrix} 2 \\ -3 \\ -5 \end{bmatrix} - 3\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ -6 \\ -10 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 3 \end{bmatrix} = \begin{bmatrix} 4 \\ -6 \\ -7 \end{bmatrix}$$

Let's use Python to double check our solution. A vector is a numpy array of a list.

```
y =  [[ 7 -4  3]]
u =  [[ 2 -3 -5]]
v =  [[0 0 1]]
```

```
array([[ 4, -6, -7]])
```

(b). To determine if $\vec{y}$ is a linear combination of $\vec{u}$ and $\vec{v}$, we need to check if there exists real numbers c_1 and c_2 such that:

$$\vec{y} = c_1\vec{u} + c_2\vec{v}$$

Substituting the given values, we have:

$$\begin{bmatrix} 7 \\ -4 \\ 3 \end{bmatrix} = c_1\begin{bmatrix} 2 \\ -3 \\ -5 \end{bmatrix} + c_2\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Which is the same as:

$$\begin{bmatrix} 7 \\ -4 \\ 3 \end{bmatrix} = \begin{bmatrix} 2x_1 + 0 \\ -3c_1 + 0 \\ -5c_1 + c_2 \end{bmatrix}$$

or

$$\begin{aligned} 2c_1 &= 7 \\ -3c_1 &= -4 \\ -5c_1 + c_2 &= 3 \end{aligned}$$

This equation cannot be satisfied because there are no values of c_1 and c_2 that can make the left-hand side equal to the right-hand side. Therefore, $\vec{y}$ is not a linear combination of $\vec{u}$ and $\vec{v}$.

In Example 1b), to solve a vector equation we converted it into a linear system. In general, the vector equation

$$x_1\vec{u_1} + x_2\vec{u_1} + \cdots + x_p\vec{u_p} = \vec{y}$$

has the same solution set as the linear system whose augmented matrix is

$$[\vec{u_1}, \vec{u_2}, \ldots, \vec{u_p}|\vec{y}] \quad (*)$$

Note that for brevity, we represented the augmented matrix by its columns.

An indexed set of vectors $\{\vec{v_1}, \vec{v_2}, \ldots \vec{v_p}\}$ in $\mathbb{R}^n$ is called a **linearly independent** set if the vector equation:

$$x_1\vec{v_1} + x_2\vec{v_2} + \cdots + x_p\vec{v_p} = 0$$

has only the trivial solution. The set is called **linearly dependent** if there are real numbers $c_1, c_2, \ldots c_p$, not all zero, such that

$$c_1\vec{v_1} + c_2\vec{v_2} + \cdots + c_p\vec{v_p} = 0$$

Theorem 3

1. A set of vectors containing the zero vector is linearly dependent.
2. A set of two vectors is linearly dependent if and only if one is a multiple of the other.

Example 2

Determine if the following vectors form a linearly independent set:

$$\vec{u} = \begin{bmatrix}1\\2\\3\end{bmatrix} \quad \vec{v} = \begin{bmatrix}4\\5\\6\end{bmatrix}, \quad \vec{w} = \begin{bmatrix}7\\8\\9\end{bmatrix}$$

Solution:

We convert the augmented matrix in $(*)$ into its REF:

```
A = np.array([[1,4,7, 0], [2,5,8,0], [3,6,9,0]])
A
```

```
array([[1, 4, 7, 0],
       [2, 5, 8, 0],
       [3, 6, 9, 0]])
```

```
# Swap two rows

def swap(matrix, row1, row2):

    copy_matrix=np.copy(matrix).astype('float64')
```

(continues on next page)

(continued from previous page)

```
    copy_matrix[row1,:] = matrix[row2,:]
    copy_matrix[row2,:] = matrix[row1,:]

    return copy_matrix


# Multiple all entries in a row by a nonzero number


def scale(matrix, row, scalar):
    copy_matrix=np.copy(matrix).astype('float64')
    copy_matrix[row,:] = scalar*matrix[row,:]
    return copy_matrix

# Replacing a row1 by the sum of itself and a multiple of row2

def replace(matrix, row1, row2, scalar):
    copy_matrix=np.copy(matrix).astype('float64')
    copy_matrix[row1] = matrix[row1]+ scalar * matrix[row2]
    return copy_matrix
```

```
A1 = replace(A, 1, 0, -2)
A1
```

```
array([[ 1.,  4.,  7.,  0.],
       [ 0., -3., -6.,  0.],
       [ 3.,  6.,  9.,  0.]])
```

```
A2 = replace(A1, 2, 0, -3)
A2
```

```
array([[  1.,   4.,   7.,   0.],
       [  0.,  -3.,  -6.,   0.],
       [  0.,  -6., -12.,   0.]])
```

```
A3 = replace(A2, 2, 1, -2)
A3
```

```
array([[ 1.,  4.,  7.,  0.],
       [ 0., -3., -6.,  0.],
       [ 0.,  0.,  0.,  0.]])
```

```
A4 = scale(A3, 1, -1/3)
A4
```

```
array([[ 1.,  4.,  7.,  0.],
       [-0.,  1.,  2., -0.],
       [ 0.,  0.,  0.,  0.]])
```

Writing this matrix as a linear system, we have

$$\begin{aligned} x + y + 7z &= 0 \\ y + 2z &= 0 \end{aligned}$$

From this, it is clear that $\{\vec{u}, \vec{v}, \vec{w}\}$ is not linearly independent because for any nonzero choice of z, we get a non-trivial solution. For example, if $z = 1$, then $(-5, -2, 1)$ is a non-trivial solution.

Matrix Equation $A\vec{x} = \vec{b}$

Similarly, we can translate a linear system into a matrix equation of the form $A\vec{x} = \vec{b}$. To that end, we need to define the product on the left-hand side of equality.

Let A be an $m \times n$ matrix (a rectangular array) with columns $\vec{a_1}, \vec{a_2}, \ldots, \vec{a_n}$, and $\vec{x} \in \mathbb{R}^n$. We can define the product of A and $\vec{x}$, denoted by $A\vec{x}$, as the linear combination of the columns of A where the coefficients are the components of $\vec{x}$:

$$A\vec{x} = [\vec{a_1},\ \vec{a_2},\ \ldots,\ \vec{a_n}] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} := x_1\vec{a_1} + x_2\vec{a_2} + \cdots + x_n\vec{a_n}.$$

Note that $A\vec{x}$ is defined only if the number of columns of A equals the number of entries of $\vec{x}$

Theorem 4 (properties of the product of A and $\vec{x}$)

Suppose A is an $m \times n$ matrix, $\vec{u}$ and $\vec{v}$ are in $\mathbb{R}^n$ and c is a scalar. Then

1. $A(\vec{u} + \vec{v}) = A\vec{u} + A\vec{v}$
2. $A(c\vec{u}) = cA\vec{u}$

Now let us discuss how to represent a linear system by a matrix equation. Consider:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots a_{1n}x_n &= b_1 \quad (9.6)\\ a_{21}x_1 + a_{22}x_2 + \ldots a_{2n}x_n &= b_2 \quad (9.7)\\ &\quad (9.8)\\ a_{m1}x_1 + a_{n2}x_2 + \ldots a_{mn}x_n &= b_m \quad (9.9)\\ &\quad (9.10) \end{aligned}$$

and let

$$A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \vdots & \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix}$$

and $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ and $\vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$. The matrix A is called the **coefficient matrix**. Then we can rewrite the above linear system as: $A\vec{x} = \vec{b}$. The next theorem describes when a matrix equation has at least one solution.

Theorem 5

Let A be an $m \times n$ matrix. The following statements are logically equivalent:

1. For any $\vec{b} \in \mathbb{R}^m$, the equation $A\vec{x} = \vec{b}$ has a solution.
2. Each $\vec{b} \in \mathbb{R}^m$ is a linear combination of columns of A

3. The columns of A span $\mathbb{R}^m$
4. A has a pivot position in every row.

Example 3

Let A be a matrix whose columns are $\vec{u}, \vec{v}, \vec{w}$ from Example 2. Find vectors $\vec{b}, \vec{c} \in \mathbb{R}^3$ such that $A\vec{x} = \vec{b}$ has a solution and $A\vec{x} = \vec{c}$ does not.

Solution:

Let

$$A = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}, \quad \vec{b} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \quad \vec{c} = \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}.$$

Now, let's form the augmented matrix of equation $A\vec{x} = \vec{b}$:

$$\begin{bmatrix} 1 & 4 & 7 & 1 \\ 2 & 5 & 8 & 2 \\ 3 & 6 & 9 & 3 \end{bmatrix}$$

Now we can perform row operations to verify that it has a solution:

```
Ab = np.array([[1,4,7,1],[2,5,8,2],[3,6,9,3]])
Ab
```

```
array([[1, 4, 7, 1],
       [2, 5, 8, 2],
       [3, 6, 9, 3]])
```

```
B1 = replace(Ab, 1 , 0, -2)
B1
```

```
array([[ 1.,  4.,  7.,  1.],
       [ 0., -3., -6.,  0.],
       [ 3.,  6.,  9.,  3.]])
```

```
B2 = replace(B1, 2 , 0, -3)
B2
```

```
array([[  1.,   4.,   7.,   1.],
       [  0.,  -3.,  -6.,   0.],
       [  0.,  -6., -12.,   0.]])
```

```
B3 = replace(B2, 2, 1 , -2)
B3
```

```
array([[ 1.,  4.,  7.,  1.],
       [ 0., -3., -6.,  0.],
       [ 0.,  0.,  0.,  0.]])
```

```
B4 = scale(B3, 1 , -1/3)
B4
```

```
array([[ 1.,  4.,  7.,  1.],
       [-0.,  1.,  2., -0.],
       [ 0.,  0.,  0.,  0.]])
```

From this we see that for any $z \in \mathbb{R}$ we have a unique solution. For example if $z = 0$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

So the equation $A\vec{x} = \vec{b}$ has a solution.

Now, we solve the equation $A\vec{x} = \vec{c}$ to verify that it does not have a solution. Let's form the augmented matrix $[A|\vec{c}]$:

$$\begin{bmatrix} 1 & 4 & 7 & 1 \\ 2 & 5 & 8 & 2 \\ 3 & 6 & 9 & 4 \end{bmatrix}$$

Now we can perform row operations to verify that it does not has a solution

```
Ac = np.array([[1,4,7,1],[2,5,8,2],[3,6,9,4]])
Ac
```

```
array([[1, 4, 7, 1],
       [2, 5, 8, 2],
       [3, 6, 9, 4]])
```

```
C1 = replace(Ac, 1 , 0, -2)
C1
```

```
array([[ 1.,  4.,  7.,  1.],
       [ 0., -3., -6.,  0.],
       [ 3.,  6.,  9.,  4.]])
```

```
C2 = replace(C1, 2 , 0, -3)
C2
```

```
array([[  1.,   4.,   7.,   1.],
       [  0.,  -3.,  -6.,   0.],
       [  0.,  -6., -12.,   1.]])
```

```
C3 = scale(C2, 1 , -1/3)
C3
```

```
array([[  1.,   4.,   7.,   1.],
       [ -0.,   1.,   2.,  -0.],
       [  0.,  -6., -12.,   1.]])
```

```
C4 = replace(C3, 2 , 1, 6)
C4
```

```
array([[ 1.,  4.,  7.,  1.],
       [-0.,  1.,  2., -0.],
       [ 0.,  0.,  0.,  1.]])
```

From this, we can see that the last row corresponds to the equation:

$$0 = 1$$

Which is not possible. So $A\vec{x} = \vec{c}$ does not have a solution.

Exercises:

Exercises

1. Given the system of equations:

$$\begin{aligned} 3x + 2y &= 1 \\ x - 2y + z &= 5 \\ 2x - z &= 0 \end{aligned}$$

Write this system as a vector equation and then a matrix equation.

2. True or False?

   (a). $\vec{0} \in \text{span}\{\vec{v}_1, \vec{v}_2, \vec{v}_3\}$.

   (b). $\vec{v}_1 \in \text{span}\{\vec{v}_1, \vec{v}_2, \vec{v}_3\}$.

   (c). $\text{span}\{\vec{v}_1, \vec{v}_2\} \subseteq \text{span}\{\vec{v}_1, \vec{v}_2, \vec{v}_3\}$.

   (d). A set 4 vectors must be linearly DEPENDENT in $\mathbb{R}^3$.

9.2.4 Solution set of linear Systems.

In this section, we study and analyze the solution set of linear systems.

Homogenous Linear Systems

We start with describing the solution set of simpler linear systems where the right side of equalities are zeros. In matrix form, $A\vec{x} = \vec{0}$; such system is called **homogeneous**. Homogeneous systems are always consistent since $\vec{x} = \vec{0}$ is always a solution. We call $\vec{0}$ the trivial solution of $A\vec{x} = \vec{0}$.

Theorem 5

The homogenous system $A\vec{x} = \vec{0}$ has a nontrivial solution if and only if it has a non-pivot column.

Non-pivot columns correspond to **free variables**; all other variables can be expressed in terms of free variables.

Example 1

Describe the solution set of $10x - 4y + 7z = 0$

Solution: consider the augmented matrix of the system: $\begin{bmatrix}10 & -4 & 7 & 0\end{bmatrix}$. There is only one pivot column and y and z are free variables.

$$\vec{x} = \begin{bmatrix}x\\y\\z\end{bmatrix} = \begin{bmatrix}0.4y - 0.7z\\y\\z\end{bmatrix} = \begin{bmatrix}0.4y\\y\\0\end{bmatrix} + \begin{bmatrix}-0.7z\\0\\z\end{bmatrix} = y\begin{bmatrix}0.4\\1\\0\end{bmatrix} + z\begin{bmatrix}-0.7\\0\\1\end{bmatrix}$$

Thus, the solution set is span ($\begin{bmatrix}0.4\\1\\0\end{bmatrix}, \begin{bmatrix}-0.7\\0\\1\end{bmatrix}$)

Example 2

Determine if the following system has a nontrivial solution. Then describe the solution set.

$$\begin{aligned} 3x + 5y - 4z &= 0\\ -3x - 2y + 4z &= 0\\ 6x + y - 8z &= 0 \end{aligned}$$

We first set up the augmented matrix, and then, we find its REF:

```
import numpy as np

A= np.array([[3,5,-4, 0], [-3, -2, 4, 0], [6, 1, -8, 0]])
A
```

```
array([[ 3,  5, -4,  0],
       [-3, -2,  4,  0],
       [ 6,  1, -8,  0]])
```

```
# Swap two rows

def swap(matrix, row1, row2):

    copy_matrix=np.copy(matrix).astype('float64')

    copy_matrix[row1,:] = matrix[row2,:]
    copy_matrix[row2,:] = matrix[row1,:]

    return copy_matrix


# Multiple all entries in a row by a nonzero number


def scale(matrix, row, scalar):
    copy_matrix=np.copy(matrix).astype('float64')
    copy_matrix[row,:] = scalar*matrix[row,:]
    return copy_matrix

# Replacing a row1 by the sum of itself and a multiple of row2

def replace(matrix, row1, row2, scalar):
    copy_matrix=np.copy(matrix).astype('float64')
    copy_matrix[row1] = matrix[row1]+ scalar * matrix[row2]
    return copy_matrix
```

```
A_1 = replace(A, 1,0, 1)
A_1
```

```
array([[ 3.,  5., -4.,  0.],
       [ 0.,  3.,  0.,  0.],
       [ 6.,  1., -8.,  0.]])
```

```
A_2 = replace(A_1, 2,0, -2)
A_2
```

```
array([[ 3.,  5., -4.,  0.],
       [ 0.,  3.,  0.,  0.],
       [ 0., -9.,  0.,  0.]])
```

```
A_3 = replace(A_2, 2, 1, 3)
A_3
```

```
array([[ 3.,  5., -4.,  0.],
       [ 0.,  3.,  0.,  0.],
       [ 0.,  0.,  0.,  0.]])
```

We can see that the third column is a non-pivot column and therefore z is a free variable. Thus, $A\vec{x} = 0$ has (infinitely many) one for each choice of z. To find the solution set we write back the linear system corresponding to REF of A:

$$\begin{aligned} 3x + 5y - 4z &= 0 \\ 3y &= 0 \\ 0 &= 0 \end{aligned}$$

Therefore, $x = \frac{4}{3}z$, and the solution set contains vectors of the following form:

$$\vec{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{4}{3}z \\ 0 \\ z \end{bmatrix} = z \begin{bmatrix} \frac{4}{3} \\ 0 \\ 1 \end{bmatrix}$$

for $z \in \mathbb{R}$.

In other words, the solution set is span $\left(\begin{bmatrix} \frac{4}{3} \\ 0 \\ 1 \end{bmatrix} \right)$

We can represent a linear independence equation

$$x_1\vec{v_1} + x_2\vec{v_2} + \cdots + x_p\vec{v_p} = 0$$

by a homogenous system $A\vec{x} = 0$ where A is a matrix whose columns are $\vec{v_i}s$. In fact, we have:

Theorem 6

The columns of a matrix A form a linearly independent set if and only if $A\vec{x} = 0$ has only trivial solution.

Example 3

Determine if the columns of the matrix $A = \begin{bmatrix} 2 & 3 & 5 \\ -1 & 1 & 1 \\ 5 & 4 & 5 \end{bmatrix}$ form a linearly independent set.

Solution:

We show $A\vec{x} = 0$ has a nontrivial solution. Therefore, by Theorem 3, the columns of A are linearly independent.

```
A = np.array([[2,3,5,0], [-1,1,1,0], [5, 4, 5,0]])
A
```

```
array([[ 2,  3,  5,  0],
       [-1,  1,  1,  0],
       [ 5,  4,  5,  0]])
```

```
A1 = swap(A, 1,0)
A1
```

```
array([[-1.,  1.,  1.,  0.],
       [ 2.,  3.,  5.,  0.],
       [ 5.,  4.,  5.,  0.]])
```

```
A2 = scale(A1 , 0, -1)
A2
```

```
array([[ 1., -1., -1., -0.],
       [ 2.,  3.,  5.,  0.],
       [ 5.,  4.,  5.,  0.]])
```

```
A3 = replace(A2, 1,0,-2)
A3
```

```
array([[ 1., -1., -1., -0.],
       [ 0.,  5.,  7.,  0.],
       [ 5.,  4.,  5.,  0.]])
```

```
A4 = replace(A3, 2,0,-5)
A4
```

```
array([[ 1., -1., -1., -0.],
       [ 0.,  5.,  7.,  0.],
       [ 0.,  9., 10.,  0.]])
```

```
A5 = scale(A4, 1, 1/5)
A5
```

```
array([[ 1. , -1. , -1. , -0. ],
       [ 0. ,  1. ,  1.4,  0. ],
       [ 0. ,  9. , 10. ,  0. ]])
```

```
A6 = replace(A5, 2,1, -9)
A6
```

```
array([[ 1. , -1. , -1. , -0. ],
       [ 0. ,  1. ,  1.4,  0. ],
       [ 0. ,  0. , -2.6,  0. ]])
```

Representing this as a linear system, the last row of A_6 shows that $z = 0$, and therefore, $y = x = 0$. In other words, the only solution to this system is zero.

General Case: the solution set of non-homogenous linear system:

Theorem 7

Let $A\vec{x} = \vec{b}$ be a consistent system, and let $\vec{p}$ be a solution. Then the solution set of $A\vec{x} = \vec{b}$ is the set of all vectors of the form $\vec{p} + \vec{h}$ where $\vec{h}$ is any solution to the homogenous system $A\vec{x} = 0$.

Example 4

Let $A\vec{x} = \vec{b}$ be a consistent system, and let $\vec{p}$ be a solution. Then the solution set of $A\vec{x} = \vec{b}$ is the set of all vectors of the form $\vec{p} + \vec{h}$ where $\vec{h}$ is any solution to the homogenous system $A\vec{x} = 0$.

Solution :

```
# set up the augmented matrix [A|b]
A = np.array([[3,5,-4,7], [-3,-2, 4,-1], [6,1,-8,-4]])
A
```

```
array([[ 3,  5, -4,  7],
       [-3, -2,  4, -1],
       [ 6,  1, -8, -4]])
```

```
# Row reduce the augmented matrix to reduced echelon form:

A1 = replace(A, 1,0,1)
A1
```

```
A2 = replace(A1, 2, 0, -2)
A2
```

```
A3 = replace(A2, 2, 1, 3)
A3
```

```
A4 = replace(A3, 0, 1, -5/3)
A4
```

```
array([[  1.        ,  -9.33333333, -12.66666667,  -0.        ],
       [  0.        ,   5.        ,   7.        ,   0.        ],
       [  5.        ,   4.        ,   5.        ,   0.        ]])
```

```
A5 = scale(A4, 0, 1/3)
A5
```

```
array([[ 0.33333333, -3.11111111, -4.22222222, -0.        ],
       [ 0.        ,  5.        ,  7.        ,  0.        ],
       [ 5.        ,  4.        ,  5.        ,  0.        ]])
```

```
A6 = scale(A5, 1, 1/3)
A6
```

```
array([[ 0.33333333, -3.11111111, -4.22222222, -0.        ],
       [ 0.        ,  1.66666667,  2.33333333,  0.        ],
       [ 5.        ,  4.        ,  5.        ,  0.        ]])
```

Now we can express each dependent variable in terms of free variables:

We have $x_1 - \frac{4}{3}x_3 = -1$, $x_2 = 2$, and x_3 is free. As a vector, the general solution of $A\vec{x} = \vec{b}$ is

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 + \frac{4}{3}x_3 \\ 2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 0 \end{bmatrix} + x_3 \begin{bmatrix} \frac{4}{3} \\ 0 \\ 1 \end{bmatrix}$$

Exercises

1. Is

$$\left\{ \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}, \begin{bmatrix} 7 \\ 8 \\ 9 \end{bmatrix} \right\}$$

linearly independent or dependent?

2. Describe the solution set of $A\vec{x} = \vec{b}$ where

$$A = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}, \quad \vec{b} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix},$$

```
import numpy as np
```

9.3 Matrices and Determinants

9.3.1 Matrix Algebra

This section focuses on matrix algebra and its properties. We will also explore how operations involving matrices are connected to linear systems.

Matrices

Consider matrix A, which is an $m \times n$ matrix with m rows and n columns:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

The entry in the ith row and jth column is referred to as the (i, j)**-entry** of matrix A. Entries with the same row and column index ($a_{i,i}$ for any i) are called **diagonal entries** of A. The ith column of A, denoted as a_i, represents a vector in $\mathbb{R}^m$. We can represent matrix A in a compact form by its columns:

$$A = \begin{bmatrix} a_1 & a_2 & \dots & a_n \end{bmatrix},$$

or by its entries:

$$A = (a_{i,j})$$

A matrix whose entries are all zero is called a **zero matrix**, denoted by 0. The size of a zero matrix is usually clear from the context.

An $n \times n$ matrix is called:

1. the **identity matrix**, denoted by I_n, if it has 1 on the diagonal and 0 everywhere else.
2. a **diagonal matrix** if it has all nondiagonal entries equal to zero. Identity matrices are diagonal.
3. an **upper triangular matrix** if it has all entries below the diagonal equal to zero.
4. a **lower triangular matrix** if it has all entries above the diagonal equal to zero.
5. a **triangular matrix** if it is either upper triangular or lower triangular.

Example 1

Let

$$A = \begin{bmatrix} 0 & 2 & -1 \\ 2 & 3 & 1 \end{bmatrix}.$$

As we have seen before, we use NumPy arrays to represent A. Note that each list in the array represents a row of A.

```
import numpy as np

A = np.array([[0, 2, -1],[2, 3, 1]])
print('A =\n', A)
```

```
A =
 [[ 0  2 -1]
 [ 2  3  1]]
```

A is a 2×3 matrix. The following cell shows how to find the size of A:

```
#Size of A

print('A has ', A.shape[0],  'rows, and', A.shape[1],  'columns')
```

```
A has  2 rows, and 3 columns
```

We can also print rows and columns of A:

```
print('The first row of A is ', A[0,:])

print(30*'*')

print('\nThe second row of A is ', A[1,:])

print(30*'*')

print('\nThe first column of A is ', A[:,0])

print(30*'*')

print('\nThe second column of A is ', A[:,1])

print(30*'*')

print('\nThe third column of A is ', A[:,2])
```

```
The first row of A is  [ 0  2 -1]
******************************

 The second row of A is  [2 3 1]
******************************

 The first column of A is  [0 2]
******************************

 The second column of A is  [2 3]
******************************

 The third column of A is  [-1  1]
```

The following cell shows how to generate an identity matrix of an arbitrary size:

```
# 3x3 identity matrix
I_3 = np.eye(3)

print('I_3 = \n \n', I_3)

# 4x4 identity matrix I_4
I_4 = np.eye(4)

print('\n I_4 = \n \n', I_4)

# 5x5 identity matrix I_5
I_5 = np.eye(5)

print('\n I_5 = \n \n', I_5)
```

```
I_3 =

 [[1. 0. 0.]
 [0. 1. 0.]
```

(continues on next page)

(continued from previous page)

```
 [0. 0. 1.]]

 I_4 =

 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]

 I_5 =

 [[1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1.]]
```

Let's have more examples:

```
B = np.array([[1, 2, 3],[4, 5, 6]])
print('B =\n', B)


#Size of B

print('\n B has ', B.shape[0],  'rows, and', B.shape[1],  'columns')

#The second column of B

print('\n The second column of B is ', B[:,1])

print(30*'*')

C = np.array([[0, 2] ,[2, 3],[5, -2]])
print('\n C = \n', C)


#Size of C

print('\n C has ', C.shape[0],  'rows, and', C.shape[1],  'columns')


#Print the columns of C

for i in range(C.shape[1]):
    print('\n The', i+1, '-th column of C is ', B[:,i])
```

```
B =
 [[1 2 3]
 [4 5 6]]

 B has  2 rows, and 3 columns

 The second column of B is  [2 5]
******************************
```

(continues on next page)

(continued from previous page)

```
C =
[[ 0  2]
 [ 2  3]
 [ 5 -2]]

C has  3 rows, and 2 columns

The 1 -th column of C is  [1 4]

The 2 -th column of C is  [2 5]
```

Basic Algebraic Operations

The arithmetic operations defined for vectors can be extended to matrices:

1. Two matrices A and B are equal if they have the same size and entries.
2. The **sum** of matrices is defined for matrices of the same size:

let $A = (a_{ij})$ and $B = (b_{ij})$ be two $m \times n$ matrices. Then $A + B$ is an $m \times n$ matrix whose entries are the sums of the corresponding entries of A and B:

$$A + B = (a_{ij} + b_{ij}).$$

3. The **scalar product** cA for a scalar $c \in \mathbb{R}$ and a matrix A is a matrix of the same size as A whose entries are c times the entries of A:

$$cA = (ca_{ij})$$

Theorem 1 (properties of sum and scalar product)

Let A, B, and C have the same size, and let c and d be real numbers.

1. $A + B = B + A$
2. $(A + B) + C = A + (B + C)$
3. $A + 0 = A$
4. $c(A + B) = cA + cB$
5. $(c + d)A = cA + dA$
6. $c(dA) = (cd)A$

Example 2

Consider the matrices in Example 1. Compute the following:

1. $3A - B$
2. $A + C$

Solution:

```
#1

3*A-B
```

```
array([[-1,  4, -6],
       [ 2,  4, -3]])
```

$A + C$ is not defined because A and C are not of the same size:

```
#2.
A + C
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[8], line 2
      1 #2.
----> 2 A + C

ValueError: operands could not be broadcast together with shapes (2,3) (3,2)
```

Matrix Product

Let A be an $m \times n$ and B be an $n \times p$ matrix. The product AB is an $m \times p$ matrix whose columns are $Ab_1, Ab_2, \ldots, Ab_p$:

$$AB = \begin{bmatrix} Ab_1 & Ab_2 & \ldots & Ab_n \end{bmatrix}.$$

We can also describe the matrix product by the entries: the (ij)-entry of AB is the product of the ith-row of A and the jth column of B in the following way:

$$a_i b_j = \begin{bmatrix} a_{i1} & a_{i2} & \ldots & a_{in} \end{bmatrix} \begin{bmatrix} b_{1j} & b_{2j} & \ldots & a_{nj} \end{bmatrix} = a_{i1}b_{j1} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj} = \Sigma_{k=1}^{n} a_{ik}b_{kj}$$

The product BA is not defined if $p \neq m$ (the *neighboring dimensions* do not match).

Example 3

Let

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \text{ and } B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}.$$

Compute AB and BA.

Solution:

A is a 2×3 and B is a 3×2 matrix; so the product is defined. In Python, we use the symbol `@` to compute the matrix product

```
#A
A = np.array([[1,2,3],[4,5,6]])

#B
B = np.array([[1,2],[3,4],[5,6]])
```

(continues on next page)

(continued from previous page)

```
#Compute the matrix product
AB = A @ B

print(AB)
```

```
[[22 28]
 [49 64]]
```

Now let's compute BA:

```
#Compute the product BA
BA = B @ A

print(BA)
```

```
[[ 9 12 15]
 [19 26 33]
 [29 40 51]]
```

Example 4

Let

$$C = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 5 & 6 & 7 \end{bmatrix}.$$

The matrix product AC is not defined because the number of columns in matrix A does not match the number of rows in matrix C. If we proceed with computing AC we will get:

```
C = np.array([[1, 2, 3, 4], [4, 5, 6, 7]])

# Compute AC
AC = A @ C

print(AC)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[11], line 4
      1 C = np.array([[1, 2, 3, 4], [4, 5, 6, 7]])
      3 # Compute AC
----> 4 AC = A @ C
      6 print(AC)

ValueError: matmul: Input operand 1 has a mismatch in its core dimension 0, with␣
 ↪gufunc signature (n?,k),(k,m?)->(n?,m?) (size 2 is different from 3)
```

Power of Square Matrices

We can raise a square matrix A to the power of a natural number using matrix multiplication. The matrix power is a fundamental idea in several linear algebra applications such as the long-term behavior of discrete dynamical systems.

Example 5

Let $M = BA$ where A and B are matrices in Example 3. Compute M^4.

```
#C
M = BA
#Compute C^4
M4 = M @ M @ M @ M

print('C^4 =\n ', M4 )
```

```
C^4 =
  [[ 5447976  7470048  9492120]
 [11879096 16288144 20697192]
 [18310216 25106240 31902264]]
```

Interestingly, computing the power of diagonal matrices is relatively straightforward.

Example 6

Let's consider the matrix $A = \begin{bmatrix} 2 & 0 \\ 0 & 5 \end{bmatrix}$. We can observe the pattern for computing powers of A:

- $A^2 = \begin{bmatrix} 2^2 & 0 \\ 0 & 5^2 \end{bmatrix} = \begin{bmatrix} 4 & 0 \\ 0 & 25 \end{bmatrix}$
- $A^3 = \begin{bmatrix} 2^3 & 0 \\ 0 & 5^3 \end{bmatrix} = \begin{bmatrix} 8 & 0 \\ 0 & 125 \end{bmatrix}$
- $A^4 = \begin{bmatrix} 2^4 & 0 \\ 0 & 5^4 \end{bmatrix} = \begin{bmatrix} 16 & 0 \\ 0 & 625 \end{bmatrix}$

We can observe that A^k is a diagonal matrix of the same size with the diagonal elements being obtained by raising the corresponding diagonal elements of A to the power of k.

```
A = np.array([[2,0],[0,5]])

A2 = A @ A
A2
```

```
array([[ 4,  0],
       [ 0, 25]])
```

```
A3 = A2 @ A
A3
```

```
array([[  8,   0],
       [  0, 125]])
```

Theorem 2 (Properties of matrix product)

Let A be an $m \times n$ matrix, and let B and C be matrices of the same size for which the following sum and product operations are defined:

1. $A(BC) = (AB)C$
2. $A(B + C) = AB + AC$
3. $(B + C)A = BA + CA$
4. $r(AB) = (rA)B = A(rB)$ for any scalar $r \in \mathbb{R}$.
5. $I_m A = AI_n = A$

Recall that I_n is the $n \times n$ identity matrix.

Numerical Note

NumPy provides two other methods (built-in functions) to compute the matrix product: `numpy.dot` and `numpy.matmul`.

They are both used to compute the product of two arrays (e.g., vectors, matrices, and even higher dimensional).

(1). If A and B are vectors (1-D arrays), they both perform the dot product of vectors (which we will define in section 6)

(2). If A and B are non-vector matrices (2-D arrays), they both perform matrix multiplication, but using `numpy.matmul(A,B)` or `A @ B` is preferred.

(3) For scalar $r \in \mathbb{R}$ `numpy.dot(r,B)` is scalar product `r*B`. But `numpy.malmal(r,B)` and `r @ B` are not defined

Example 7

Let A, B, and C be the matrices defined in Example 3. Let's explore the differences between the above operations:

```
import numpy as np

#A
A = np.array([[1,2,3],[4,5,6]])

#B
B = np.array([[1,2],[3,4],[5,6]])


# for matrices (2D-arrays) of compatible size all methods return matrix product

print('\n A@B = \n', A@B)

print(10*'*')

print('\n np.dot(A,B) = \n', np.dot(A,B))

print(10*'*')

print('\n np.matmul(A,B) = \n', np.matmul(A,B))
```

```
 A@B =
 [[22 28]
 [49 64]]
**********
```

(continues on next page)

(continued from previous page)

```
 np.dot(A,B) =
 [[22 28]
 [49 64]]
**********

 np.matmul(A,B) =
 [[22 28]
 [49 64]]
```

```
# for vectors (1D arrays) dot and matmul return the dot product
C = [1,2,3]
D = [4,5,6]

print('\n np.dot(C,D) = \n', np.dot(C,D))

print(10*'*')

print('\n np.matmul(C,D) = \n', np.matmul(C,D))
```

```
 np.dot(C,D) =
 32
**********

 np.matmul(C,D) =
 32
```

```
# C@D is not defined for vectors

print('\n A@B = \n', C @ D)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[17], line 3
      1 # C@D is not defined for vectors
----> 3 print('\n A@B = \n', C @ D)

TypeError: unsupported operand type(s) for @: 'list' and 'list'
```

```
# If one argument is a real number, dot returns the scalar product
print('\n np.matmul(2,B) = \n', np.dot(A,2))
```

```
 np.matmul(2,B) =
 [[ 2  4  6]
 [ 8 10 12]]
```

```
# If one argument is a real number  @ and matmul are not defined
2 @ B
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[19], line 2
      1 # If one argument is a real number  @ and matmul are not defined
----> 2 2 @ B
```

(continues on next page)

(continued from previous page)

```
ValueError: matmul: Input operand 0 does not have enough dimensions (has 0, gufunc␣
 ↪core with signature (n?,k),(k,m?)->(n?,m?) requires 1)
```

```
np.matmul (2,B)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[20], line 1
----> 1 np.matmul (2,B)

ValueError: matmul: Input operand 0 does not have enough dimensions (has 0, gufunc␣
 ↪core with signature (n?,k),(k,m?)->(n?,m?) requires 1)
```

Exercises

1. Which pair of the following matrices can be multiplied? Compute their matrix product.

$$A = \begin{bmatrix} 1 & 3 \\ 4 & -2 \\ 3 & 2 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 4 & 5 \\ 0 & 2 & 3 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 3 & 0 \\ 2 & -1 & 3 \\ 0 & 1 & 1 \end{bmatrix}$$

2. Find two matrices A and B such that the products AB and BA are defined but $AB \neq BA$.
3. Given

$$A = \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix},$$

find a nonzero matrix C for which

$$AC = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

9.3.2 Transpose, Inverse and Trace of matrices

In this section, we will continue studying the basic operators on the set of matrices, focusing on the transpose, inverse, and trace of a square matrix.

Transpose of a matrix

Given an $m \times n$ matrix A, the transpose of A is the $n \times m$ matrix denoted by A^T, whose columns are the corresponding rows of A.

Example 1

If

$$M = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix},$$

then

$$M^T = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}.$$

In Python, we can either use `narray.T` or `narray.transpose()` to find the transpose of a matrix.

```
import numpy as np

M = np.array([[1,2],[3, 4],[5,6]])

#Compute the the transpose of M
M_T = M.T

print("M = ", '\n', M, '\n\n', "M_T = ",'\n', M_T)
```

```
M =
 [[1 2]
 [3 4]
 [5 6]]

 M_T =
 [[1 3 5]
 [2 4 6]]
```

```
#Compute the the transpose of M
M_transpose = M.transpose()

print("M = ", '\n', M, '\n\n', "M_T = ",'\n', M_transpose)
```

```
M =
 [[1 2]
 [3 4]
 [5 6]]

 M_T =
 [[1 3 5]
 [2 4 6]]
```

Note that `transpose` and `T` both have different effect on a list and a list of lists:

```
# transpose of a list of lists

np.array([[1,2,3]]).transpose()
```

```
array([[1],
       [2],
       [3]])
```

```
# does not effect 1D array
np.array([1,2,3]).transpose()
```

```
array([1, 2, 3])
```

Inverse of a square matrix

An $n \times n$ matrix A is called invertible if there exists an $n \times n$ matrix B such that $AB = BA = I_n$. In this case, B is referred to as the inverse of A, denoted as A^{-1}. It is important to note that not all matrices are invertible.

Theorem 3

Let $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$. If $ad - bc \neq 0$, then A is invertible, and its inverse is given by:

$$A^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

$ad - bc$ is called the **determinant** of matrix A. The determinant plays a crucial role in determining the invertibility of a square matrix. If the determinant is zero, i.e., $ad - bc = 0$, then matrix A is not invertible. Determinants of matrices will be discussed in the next section, where we will explore their properties and computation methods.

Example 2

1. Find the inverse of $M = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$.
2. Find the inverse of $N = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$.

Solution:

Let's write a Python code that computes the inverse of a 2×2 matrix:

```
M = np.array([[1, 2], [3, 4]])
N = np.array([[1, 2], [2, 4]])

# Compute the inverse
def inverse_matrix(matrix):
    a, b = matrix[0]
    c, d = matrix[1]
```

(continues on next page)

(continued from previous page)

```
    det = a * d - b * c

    if det == 0:
        print(matrix, "is not invertible")
    else:
        D = 1 / det
        inverse = np.zeros((2, 2))
        inverse[0, 0] = D * d
        inverse[0, 1] = -D * b
        inverse[1, 0] = -D * c
        inverse[1, 1] = D * a

        return(inverse)
```

```
inverse_matrix(M)
```

```
array([[-2. ,  1. ],
       [ 1.5, -0.5]])
```

```
inverse_matrix(N)
```

```
[[1 2]
 [2 4]] is not invertible
```

Theorem 4 (properties of the inverse matrix)

Let A and B be two invertible $n \times n$ matrices.

1. $(A^{-1})^{-1} = A$
2. $(AB)^{-1} = B^{-1}A^{-1}$
3. $(A^T)^{-1} = (A^{-1})^T$

Theorem 5 (computing the inverse of a square matrix)

An $n \times n$ matrix A is invertible if and only if it is row equivalent to the identity matrix I_n.

We can use this theorem to find the inverse of a matrix using the row reduction method: set up the augmented matrix $[A|I_n]$ and perform row operations until the left side of the augmented matrix becomes the identity matrix I_n. If this is achieved, then the right side of the augmented matrix will be the inverse of A. Otherwise, if the left side does not become I_n, then the matrix A is not invertible.

Example 3

Find the inverse of $M = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 3 \\ 4 & -3 & 8 \end{bmatrix}$.

Solution Lets set up the augmented matrix and find its RREF:

$$[M|I_3] = \begin{bmatrix} 0 & 1 & 2 & 1 & 0 & 0 \\ 1 & 0 & 3 & 0 & 1 & 0 \\ 4 & -3 & 8 & 0 & 0 & 1 \end{bmatrix}$$

```
# matrix M

M = np.array([[0,1,2], [1,0,3], [4, -3, 8]])
M
```

```
array([[ 0,  1,  2],
       [ 1,  0,  3],
       [ 4, -3,  8]])
```

```
# identity matrix I_3

I3 = np.eye(3)
I3
```

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

```
# Augmented matrix [A,I_3]

A = np.concatenate((M, I3), axis = 1)
A
```

```
array([[ 0.,  1.,  2.,  1.,  0.,  0.],
       [ 1.,  0.,  3.,  0.,  1.,  0.],
       [ 4., -3.,  8.,  0.,  0.,  1.]])
```

Now lets call row operations

```
# Elementry row operations:

# Swap two rows

def swap(matrix, row1, row2):

    copy_matrix=np.copy(matrix).astype('float64')

    copy_matrix[row1,:] = matrix[row2,:]
    copy_matrix[row2,:] = matrix[row1,:]

    return copy_matrix


# Multiple all entries in a row by a nonzero number


def scale(matrix, row, scalar):
    copy_matrix=np.copy(matrix).astype('float64')
    copy_matrix[row,:] = scalar*matrix[row,:]
    return copy_matrix
```

(continues on next page)

(continued from previous page)

```
# Replacing a row by the sum of itself and a multiple of another

def replace(matrix, row1, row2, scalar):
    copy_matrix=np.copy(matrix).astype('float64')
    copy_matrix[row1] = matrix[row1]+ scalar * matrix[row2]
    return copy_matrix
```

```
A_1 = swap(A, 0, 1)
A_1
```

```
array([[ 1.,  0.,  3.,  0.,  1.,  0.],
       [ 0.,  1.,  2.,  1.,  0.,  0.],
       [ 4., -3.,  8.,  0.,  0.,  1.]])
```

```
A_2 = replace(A_1, 2, 0, -4)
A_2
```

```
array([[ 1.,  0.,  3.,  0.,  1.,  0.],
       [ 0.,  1.,  2.,  1.,  0.,  0.],
       [ 0., -3., -4.,  0., -4.,  1.]])
```

```
A_3 = replace(A_2, 2, 1, 3)
A_3
```

```
array([[ 1.,  0.,  3.,  0.,  1.,  0.],
       [ 0.,  1.,  2.,  1.,  0.,  0.],
       [ 0.,  0.,  2.,  3., -4.,  1.]])
```

```
A_4 = scale(A_3, 2, 1/2)
A_4
```

```
array([[ 1. ,  0. ,  3. ,  0. ,  1. ,  0. ],
       [ 0. ,  1. ,  2. ,  1. ,  0. ,  0. ],
       [ 0. ,  0. ,  1. ,  1.5, -2. ,  0.5]])
```

```
A_5 = replace(A_4, 1, 2, -2)
A_5
```

```
array([[ 1. ,  0. ,  3. ,  0. ,  1. ,  0. ],
       [ 0. ,  1. ,  0. , -2. ,  4. , -1. ],
       [ 0. ,  0. ,  1. ,  1.5, -2. ,  0.5]])
```

```
A_6 = replace(A_5, 0, 2, -3)
A_6
```

```
array([[ 1. ,  0. ,  0. , -4.5,  7. , -1.5],
       [ 0. ,  1. ,  0. , -2. ,  4. , -1. ],
       [ 0. ,  0. ,  1. ,  1.5, -2. ,  0.5]])
```

We can see that the resulting matrix is in the form $[I_3, B]$. Thus, $B = \begin{bmatrix} -4.5 & 7 & -1.5 \\ -2 & 4 & -1 \\ 1.5 & -2 & 0.5 \end{bmatrix}$ is the inverse of M.

Remark We can also use an `np.linalg.inv()` function, from NumPy linear algebra submodule, to find an inverse of a matrix:

```
np.linalg.inv(M)
```

```
array([[-4.5,  7. , -1.5],
       [-2. ,  4. , -1. ],
       [ 1.5, -2. ,  0.5]])
```

Theorem 6 (The Invertible Matrix Theorem)

Let A be an $n \times n$ matrix. The following statements are equivalent:

1. A is an invertible matrix.
2. A is row equivalent to I_n.
3. The equation $A\vec{x} = 0$ has only the trivial solution ($\vec{x} = \mathbf{0}$).
4. The columns of A form a linearly independent set.
5. The equation $A\vec{x} = \vec{b}$ has a unique solution ($\vec{x} = A^{-1}\vec{b}$).
6. The columns of A span $\mathbb{R}^n$.

Example 4

Find the solution set of the following linear systems:

1. $M\vec{x} = 0$, where M is the matrix from Example 7.
2. $M\vec{x} = \vec{b}$, where M is the matrix from Example 7 and $\vec{b} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$.

Solution:

1. Since M is invertible, by the Invertible Matrix Theorem part (3), the only solution to $M\vec{x} = 0$ is the zero vector $\vec{x} = \mathbf{0}$.
2. By IMT part (5), the only solution to $M\vec{x} = \vec{b}$ is $\vec{x} = M^{-1}\vec{b}$. In Example 7, we computed the inverse of M:

$$A^{-1} = \begin{bmatrix} -4.5 & 7 & -1.5 \\ -2 & 4 & -1 \\ 1.5 & -2 & 0.5 \end{bmatrix}$$

In the following cell, we use B to represent A^{-1} to simplify notation:

```
B = np.array([[-4.5, 7, -1.5], [-2, 4, -1], [1.5, -2, 0.5]])

b = np.array ([[1], [2], [3]])

#compute Bb

x = B @ b
x
```

```
array([[ 5.],
       [ 3.],
       [-1.]])
```

```
M = np.array([[0,1,2], [1,0,3], [4,-3,8]])

#compute Mx

Mx= M @ x
Mx
```

```
array([[1.],
       [2.],
       [3.]])
```

which is $\vec{b}$.

Trace of Square Matrices

The trace of an $n \times n$ matrix A, denoted by $Tr(A)$ or $tr(A)$, is defined as the sum of the elements on its main diagonal:

$$Tr(A) = a_{11} + a_{22} + \cdots + a_{nn} = \sum_{i=1}^{n} a_{ii}$$

Theorem 7 (properties of the trace)

Let A and B be two $n \times n$ matrices, and $c \in \mathbb{R}$. Then we have the following:

1. **Trace of sum:** $Tr(A + B) = Tr(A) + Tr(B)$
2. **Trace of scalar product:** $Tr(cA) = c \cdot Tr(A)$
3. **Trace of identity matrix:** $Tr(I_n) = n$
4. **Trace of product:** $Tr(AB) = Tr(BA)$

The trace can also be seen as a function from the set of $n \times n$ matrices to real numbers. It can be shown that there is only one function that satisfies the above conditions.

Example 5

Let $M = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 3 \\ 4 & -3 & 8 \end{bmatrix}$.

Then: $Tr(M) = 0 + 0 + 8 = 8$.

For large matrices, we can use `numpy.trace()` to compute the trace. The following code generates a random 5×5 matrix and computes its trace:

```
# generate a 5x5 matrix
A = np.random.rand(5, 5)
```

(continues on next page)

(continued from previous page)

```
print(A)


#compute the trace
Tr_A = np.trace(A)
print('\n Tr(A) = ', Tr_A)
```

```
[[0.34123286 0.3788517  0.62749816 0.49779428 0.85625537]
 [0.49841636 0.64059001 0.97192421 0.82065731 0.73171872]
 [0.93029465 0.82685801 0.34255562 0.22621385 0.40893711]
 [0.20730845 0.35221413 0.95709886 0.62781555 0.49836643]
 [0.75311689 0.22947177 0.40015054 0.20169886 0.86231891]]

 Tr(A) =  2.814512956289748
```

Exercises

1. Suppose

$$A = \begin{bmatrix} 1 & 2 \\ 0 & -1 \\ 3 & 0 \end{bmatrix}.$$

a. Compute A^T.

b. Is AA^T invertible?

c. Is $A^T A$ invertible?

2. The trace is invariant under cyclic permutations, i.e., for three matrices we have $Tr(ABC) = Tr(BCA) = Tr(CAB)$, where A is an $n \times p$ matrix, B is a $p \times m$ matrix, and C is an $m \times n$ matrix. Using Example 9, generate three matrices A, B, and C such that the product ABC makes sense, and verify that:

$$Tr(ABC) = Tr(BCA) = Tr(CAB).$$

9.3.3 Determinant

This section introduces the determinants of matrices and explores some of their properties, applications, and computation methods. The determinant of a square matrix is a useful concept in linear algebra and has many theoretical and practical roles, including:

1. Determining the invertibility of a matrix.
2. Determining the linear dependency of a set of vectors.
3. Computing the area of parallelograms or the volume of parallelepipeds.
4. Solving linear systems (Cramer's rule).
5. Finding the inverse of an invertible matrix.

Note that applications (4) and (5) are based on Cramer's rule, which is a theoretical tool but not an efficient computational method. We won't discuss Cramer's rule.

Cofactor expansion

In the previous section, we defined the determinant of a 2×2 matrix as $det(A) = ad - bc$, where $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$. To extend this definition to larger matrices, we follow a recursive approach. For example, to find the determinant of a 3×3 matrix, we can use 2×2 determinants.

Let $A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$ be a general 3×3 matrix. For any entry a_{ij}, let A_{ij} denote a 2×2 submatrix of A defined by removing the row and column of A that contain a_{ij}. For example, $A_{12} = \begin{bmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{bmatrix}$ and $A_{23} = \begin{bmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{bmatrix}$.

Example 1

Write a Python code that prints all submatrices of

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}.$$

Solution:

```
M = np.array([[1,2,3],[4,5,6],[7,8,9]])
M
```

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

```
for j in range (3):
    for i in range (3):
        submatrix = np.delete( (np.delete(M, i , 0)), j ,1)
        print (' \n M(',i,j, ') = \n \n', submatrix)
        print(30*"*")
```

```

 M( 0 0 ) =

 [[5 6]
 [8 9]]
******************************

 M( 1 0 ) =

 [[2 3]
 [8 9]]
******************************

 M( 2 0 ) =

 [[2 3]
 [5 6]]
******************************
```

(continues on next page)

(continued from previous page)

```
 M( 0 1 ) =

 [[4 6]
 [7 9]]
******************************

 M( 1 1 ) =

 [[1 3]
 [7 9]]
******************************

 M( 2 1 ) =

 [[1 3]
 [4 6]]
******************************

 M( 0 2 ) =

 [[4 5]
 [7 8]]
******************************

 M( 1 2 ) =

 [[1 2]
 [7 8]]
******************************

 M( 2 2 ) =

 [[1 2]
 [4 5]]
******************************
```

The determinant of A, expanded over the first row, is defined as:

$$det(A) = \sum_{k=1}^{3} (-1)^{1+k} a_{1k} |A_{1k}| = a_{11}|A_{11}| - a_{12}|A_{12}| + a_{13}|A_{13}|$$

Here $|A_{ij}|$ represents the determinant of the A_{ij}. Note that choosing a different row or column would give the same value. For example, using the second row:

$$det(A) = \sum_{k=1}^{3} (-1)^{2+k} a_{2k} |A_{2k}| = -a_{21}|A_{21}| + a_{22}|A_{22}| - a_{23}|A_{23}|$$

Similarly, using the second column:

$$det(A) = \sum_{k=1}^{3} (-1)^{k+2} a_{k2} |A_{k2}| = -a_{12}|A_{12}| + a_{22}|A_{22}| - a_{32}|A_{32}|$$

Example 2

Compute the determinant of

$$M = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 3 \\ 4 & -3 & 8 \end{bmatrix}.$$

Solution:

Let's write Python code that computes the determinant of a 3×3 matrix:

```
M = np.array([[0,1,2],[1,0,3],[4,-3,8]])
```

```
# Computes a 2x2 determinant
def det2(matrix):

    if matrix.shape == (2, 2):
        return matrix[0,0] * matrix[1,1] - matrix[0,1] * matrix[1,0]
    else:
        print("Please enter a 2x2 matrix")

# Computes the 3x3 determinat over a given row

def det3(matrix, row):
    if matrix.shape == (3, 3) and row in range(3):
        det = 0
        for j in range(3):
            submatrix = np.delete( (np.delete(matrix, row , 0)), j ,1)
            det += (-1) ** (row + j) * matrix[row, j] * det2(submatrix)
        return det
    else:
        print("The input matrix is not 3x3 or the row is not chosen in the right range
↪")
```

```
det3(M,0)
```

```
-2
```

We can observe that selecting a different row does not affect the value of the determinant:

```
det3(M,2)
```

```
-2
```

The determinant of an $n \times n$ matrix A can be defined the same way recursively:

$$det(A) = \sum_{i=1}^{n} (-1)^{1+i} a_{1i} det(A_{1i})$$

This formula is called **a cofactor expansion across the first row** of A.

Let's write a Python code that computes the determinant of a square matrix of an arbitrary size:

```
# Computes the determinant of an nxn matrix.

def det(matrix):
```

(continues on next page)

(continued from previous page)

```
    n = matrix.shape[0]
    if n == 1:
        return matrix[0, 0]
    elif n == 2:
        return matrix[0, 0] * matrix[1, 1] - matrix[0, 1] * matrix[1, 0]
    else:
        det = 0
        for j in range(n):
            submatrix = np.delete( (np.delete(matrix, 0 , 0)), j ,1)
            det += (-1) ** (0 + j) * matrix[0, j] * det2(submatrix)
        return det
```

```
# determinant of matrix M
det(M)
```

```
-2
```

Let's discuss the properties of determinants:

If $A = \begin{bmatrix} a & 0 \\ c & d \end{bmatrix}$, then $det(A) = ad$. We can generalize this observation using induction:

Theorem 8

The determinant of a triangular matrix is the product of its diagonal entries.

Next, we will explore the impact of row operations on determinants.

Theorem 9

1. If a multiple of one row of matrix A is added to another row to produce matrix B, then $det(A) = det(B)$.
2. If two rows are interchanged to produce matrix B, then $det(A) = -det(B)$.
3. If one row is multiplied by a scalar k to produce matrix B, then $det(A) = k \cdot det(B)$.

Consequently, if a square matrix A is reduced to U using only row replacements and r row swaps, then

$$det(A) = (-1)^r \cdot det(U)$$

By combining **Theorem 8** and **Theorem 9**, we can compute determinants effectively.

Example 3

Compute

$$det \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 3 \\ 4 & -3 & 8 \end{bmatrix}.$$

Solution:

We first convert the matrix to its row echelon form (REF) and then use Theorem 1 to find the determinant.

```
M = np.array([[0,1,2],[1,0,3],[4,-3,8]])

# Elementry row operations:

# Swap two rows

def swap(matrix, row1, row2):

    copy_matrix=np.copy(matrix).astype('float64')

    copy_matrix[row1,:] = matrix[row2,:]
    copy_matrix[row2,:] = matrix[row1,:]

    return copy_matrix


# Multiple all entries in a row by a nonzero number


def scale(matrix, row, scalar):
    copy_matrix=np.copy(matrix).astype('float64')
    copy_matrix[row,:] = scalar*matrix[row,:]
    return copy_matrix

# Replacing a row by the sum of itself and a multiple of another

def replace(matrix, row1, row2, scalar):
    copy_matrix=np.copy(matrix).astype('float64')
    copy_matrix[row1] = matrix[row1]+ scalar * matrix[row2]
    return copy_matrix
```

```
M1 = swap(M, 0, 1)
M1
```

```
array([[ 1.,  0.,  3.],
       [ 0.,  1.,  2.],
       [ 4., -3.,  8.]])
```

```
M2 = replace(M1, 2, 0, -4)
M2
```

```
array([[ 1.,  0.,  3.],
       [ 0.,  1.,  2.],
       [ 0., -3., -4.]])
```

```
M3 = replace(M2, 2, 1, 3)
M3
```

```
array([[1., 0., 3.],
       [0., 1., 2.],
       [0., 0., 2.]])
```

Since we used only one row swap, we have:

$$\det(M) = -\det(M_3)$$

Using Theorem 9, we have:

$$\det(M) = -\det(M_3) = -2$$

As mentioned in the introduction, the determinant can be used to check the invertibility of matrices.

Theorem 10 (Determinants and Invertibility)

A square matrix A is invertible if and only if $\det(A) \neq 0$.

Theorem 11

Let A and B be two square matrices of the same size. Then:

1. $\det(A^T) = \det(A)$.
2. $\det(AB) = \det(A) \cdot \det(B)$.
3. If A is invertible, then $\det(A)^{-1} = \frac{1}{\det(A)}$.

Example 4

Compute

$$\det(B^4),$$

where

$$B = \begin{bmatrix} 1 & 1 & 2 \\ 2 & 0 & 3 \\ 5 & -3 & 8 \end{bmatrix}.$$

Solution: Using Theorem 11 part 2, it's easy to see that

$$\det(B^4) = (\det(B))^4.$$

```
B = np.array([[1,1,2],[2,0,3],[5,-3,8]])

#compute det(B)
det(B)

#using Theorem 4 to compute det(B)^4

det(B) ** 4
```

```
256
```

Determinants as Area or Volume

Theorem 12

1. If A is a 2×2 matrix, then $|\det(A)|$ is the area of the parallelogram determined by the columns of A.
2. If A is a 3×3 matrix, then $|\det(A)|$ is the volume of the parallelepiped determined by the columns of A.

In particular, three points $\vec{u}, \vec{v}, \vec{w}$ in $\mathbb{R}^3$ are coplanar (i.e., they lie on the same plane) if and only if $\det([\vec{u}, \vec{v}, \vec{w}]) = 0$.

The theorem is obviously true for diagonal matrices.

Example 5

1. Let

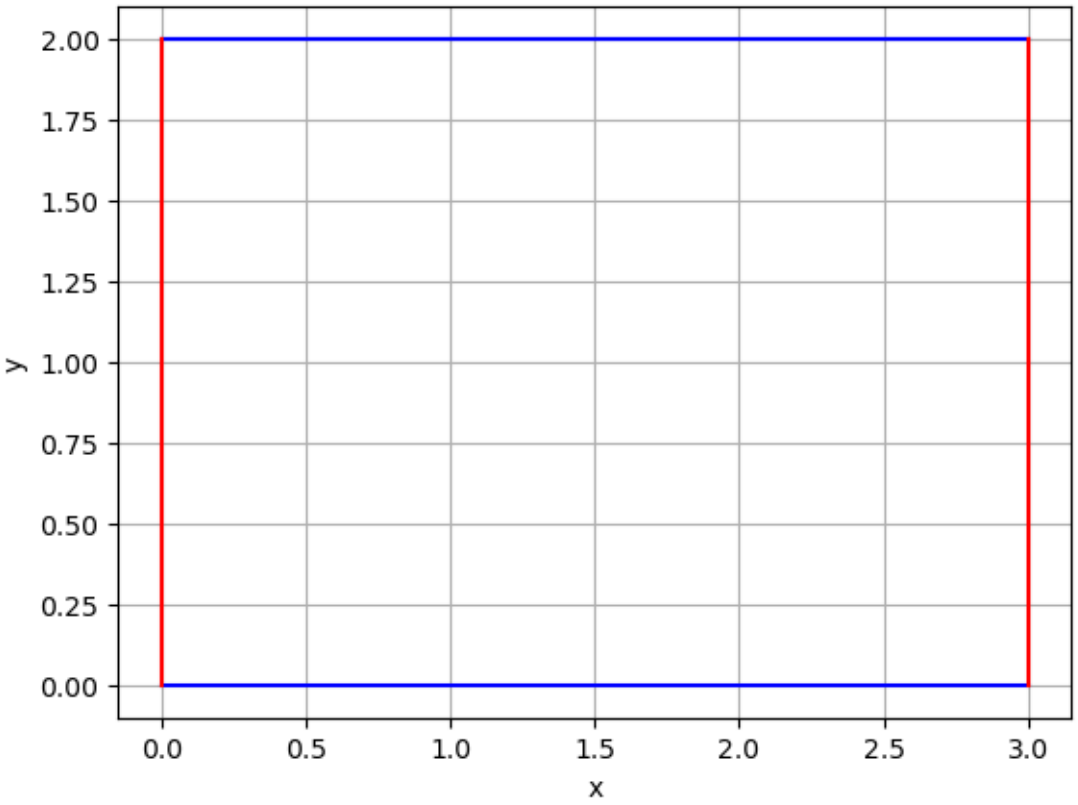
$$A = \begin{bmatrix} 3 & 0 \\ 0 & -2 \end{bmatrix}.$$

The area of the parallelogram determined by

$$\begin{bmatrix} a \\ 0 \end{bmatrix} and \begin{bmatrix} 0 \\ d \end{bmatrix}$$

is $|-6| = 6$.

```
import matplotlib.pyplot as plt

# Define x and y values for the lines
x = range(0, 4)
y1 = [ 0 for k in x]
y2 = [ 2 for k in x]
y = range(0, 3)
x1 = [3 for k in y]
x2 = [0 for k in y]

# Plot the lines on the same plane
plt.plot(x, y1, 'b')
plt.plot(x, y2, 'b')
plt.plot(x1, y, 'r')
plt.plot(x2, y, 'r')

# Add labels to the graph
plt.xlabel('x')
plt.ylabel('y')
plt.grid()

# Show the graph
plt.show()
```



Example 6

Determine if the following points are coplanar.

$$\begin{bmatrix}1\\0\\2\end{bmatrix}, \begin{bmatrix}4\\3\\5\end{bmatrix}, \begin{bmatrix}3\\4\\6\end{bmatrix}$$

Solution:

To determine if the above points are coplanar, we check whether or not $det\left(\begin{bmatrix}1 & 0 & 2\\4 & 3 & 5\\3 & 4 & 6\end{bmatrix}\right) = 0$:

```
A = np.array([[1,0,2],[4,3,5],[3,4,6]])

det(A)
```

```
12
```

Since $det(A) \neq 0$, the mentioned points are not coplanar. In fact they form a parallelepiped of volume 12.

Numerical Note

In this section, we discussed two different ways to compute the determinant of a square matrix: cofactor expansion and row operations.

While cofactor expansion is a straightforward approach, it becomes computationally impossible for large matrices. In fact, cofactor expansion requires $n!$ multiplications for an $n \times n$ matrix. If a computer can perform a trillion multiplications per second, computing the determinant of a 25×25 matrix would take approximately 500×500 years!

On the other hand, evaluating the determinant of an $n \times n$ matrix using row operations involves about $\frac{2n^3}{3}$ arithmetic operations. Using this method, the determinant of a 25×25 matrix can be computed in a fraction of a second on most microcomputers.

Exercises

1. Let A be an $n \times n$ matrix such that $A^2 = I_n$. Show that $det(A) = \pm 1$.
2. Find the determinant of

$$A = \begin{bmatrix} 1 & 1 & 2 \\ 2 & 0 & 3 \\ 5 & -3 & 8 \end{bmatrix}$$

using the row reduction method.

3. Use determinants to decide if the set

$$\left\{ \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}, \begin{bmatrix} 7 \\ 8 \\ 9 \end{bmatrix} \right\}$$

is linearly independent.

4. Suppose A, B, P are $n \times n$ matrices, P is invertible, and r is a real number. Mark True or False:
   a. $det(A + B) = det(A) + det(B)$.
   b. $det(rA) = r \cdot det(A)$.
   c. $det(PAP^{-1}) = det(A)$.
5. Determine if the following points are coplanar (i.e., they lie on the same plane):

$$\begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 4 \\ 3 \\ 5 \end{bmatrix}, \begin{bmatrix} 3 \\ 0 \\ 6 \end{bmatrix}$$

```
import numpy as np
```

9.4 Linear Transformations

9.4.1 Subspaces of $\mathbb{R}^n$

This section focuses on subsets of $\mathbb{R}^n$ that preserve $\mathbb{R}^n$'s linear structure. We will see that any matrix $m \times n$ matrix A admits two important subspaces that contain useful information about the solution set of $A\vec{x} = \vec{b}$ for some $\vec{b} \in \mathbb{R}^m$

Column space and null space

A **subspace** of $\mathbb{R}^n$ is a subset $H \subset \mathbb{R}^n$ that is closed under addition and scalar multiplication in the following sense:

1. $\vec{u} + \vec{v} \in H \quad \forall \vec{u}, \vec{v} \in H$.
2. $r\vec{u} \in H \quad \forall \vec{u} \in H$ and $\forall r \in \mathbb{R}$.

Note that we can choose $c = 0$, which implies $\vec{0}$ is an element of any subspace. The smallest subspaces of $\mathbb{R}^n$ is $H = \{\vec{0}\}$ and the largest possible subspace of $\mathbb{R}^n$ is itself. Other subspaces of $\mathbb{R}^n$ are lines, planes, and their n-dimensional counterparts ($n \geq 3$) that pass through the origin.

Example 1

Let $\vec{u}, \vec{v} \in \mathbb{R}^2$. Show that $H := \text{span}(\vec{u}, \vec{v})$ is a subspace of $\mathbb{R}^2$.

Solution:

Recall that the span of a set of vectors is the set of all possible linear combinations of those vectors. Let $\vec{x}, \vec{y} \in \text{span}(\vec{u}, \vec{v})$ and $r \in \mathbb{R}$. There exist $c_1, c_2, d_1, d_2 \in \mathbb{R}$ such that $\vec{x} = c_1\vec{u} + c_2\vec{v}$ and $\vec{y} = d_1\vec{u} + d_2\vec{v}$. We have:

$$\vec{x} + \vec{y} = (c_1 + d_1)\,\vec{u} + (c_2 + d_2)\,\vec{v} \quad \text{and} \quad r\vec{x} = (rc_1)\,\vec{u} + (rc_2)\,\vec{v}$$

Both of these vectors are in H.

In general, if $\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_p \in \mathbb{R}^n$, then the span of of $\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_p$, is a subspace $H \subset \mathbb{R}^n$, and we say H is **spanned (or generated)** by $\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_p$.

Any $m \times n$ matrix A is associated with important subspaces:

1. The **column space** of A, denoted by col(A), is the span of the columns of A:

$$col(A) = \text{span}(\{\vec{a}_1, \vec{a}_2, \ldots, \vec{a}_n\})$$

where $\vec{v}_i$s are the columns of A for $i = 1, 2, \ldots, n$. By definition, $col(A)$ is a subspace of $\mathbb{R}^m$.

2. The **null space** A, denoted by $null(A)$, is the solution set of the homogeneous equation $A\vec{x} = \vec{0}$:

$$null(A) = \{\vec{x} : \ A\vec{x} = \vec{0}\}$$

By Theorem 2 in subsection 1.2, null(A) is a subspace of $\mathbb{R}^n$.

Example 2

Let $A = \begin{bmatrix} 1 & -1 \\ 2 & 1 \\ -1 & 3 \end{bmatrix}$.

1. Find $col(A)$ and $null(A)$.
2. Is $\vec{b} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \in$ col(A)?
3. Is $\vec{u} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \in$ null(A)?

Solution:

1. $col(A) = \text{span}\left\{ \begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \\ 3 \end{bmatrix} \right\}$.

To find $null(A)$, we solve the homogeneous equation $A\vec{x} = \vec{0}$. Let's set up the augmented matrix $[A \mid \vec{0}]$ and row-reduce it:

```
M = np.array([[1, -1, 0], [2, 1, 0], [-1, 3, 0]])
M
```

```
array([[ 1, -1,  0],
       [ 2,  1,  0],
       [-1,  3,  0]])
```

```
# Swap two rows

def swap(matrix, row1, row2):

    copy_matrix=np.copy(matrix).astype('float64')

    copy_matrix[row1,:] = matrix[row2,:]
    copy_matrix[row2,:] = matrix[row1,:]

    return copy_matrix


# Multiple all entries in a row by a nonzero number


def scale(matrix, row, scalar):
    copy_matrix=np.copy(matrix).astype('float64')
    copy_matrix[row,:] = scalar*matrix[row,:]
    return copy_matrix

# Replacing a row by the sum of itself and a multiple of another

def replace(matrix, row1, row2, scalar):
    copy_matrix=np.copy(matrix).astype('float64')
    copy_matrix[row1] = matrix[row1]+ scalar * matrix[row2]
    return copy_matrix
```

```
M1 = replace(M, 1, 0, -2)
M1
```

```
array([[ 1., -1.,  0.],
       [ 0.,  3.,  0.],
       [-1.,  3.,  0.]])
```

```
M2 = replace(M1, 2, 0, 1)
M2
```

```
array([[ 1., -1.,  0.],
       [ 0.,  3.,  0.],
       [ 0.,  2.,  0.]])
```

```
M3 = scale(M2, 1, 1/3)
M3
```

```
array([[ 1., -1.,  0.],
       [ 0.,  1.,  0.],
       [ 0.,  2.,  0.]])
```

```
M4 = replace(M3, 2, 1, -2)
M4
```

```
array([[ 1., -1.,  0.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  0.]])
```

Therefore,

$$\begin{bmatrix} 1 & -1 & | & 0 \\ 2 & 1 & | & 0 \\ -1 & 3 & | & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & -1 & | & 0 \\ 0 & 1 & | & 0 \\ 0 & 0 & | & 0 \end{bmatrix}.$$

From the row-reduced form, we can see that all columns are pivots and there is no free variable. In other words, the only solution to the system is the trivial solution: $null(A) = 0$

2. $\vec{b} \in \text{col}(A)$ if and only if $\vec{b}$ is a linear combination of the columns of A. In other words, if and only if the equation $A\vec{x} = \vec{b}$ has a solution. Let's set up the augmented matrix $[A \mid \vec{b}]$ and row-reduce it:

```
import numpy as np

M = np.array([[1, -1, 1], [2, 1, 2], [-1, 3, 1]])
M
```

```
array([[ 1, -1,  1],
       [ 2,  1,  2],
       [-1,  3,  1]])
```

```
M1 = replace(M, 1, 0, -2)
M1
```

```
array([[ 1., -1.,  1.],
       [ 0.,  3.,  0.],
       [-1.,  3.,  1.]])
```

```
M2 = replace(M1, 2, 0, 1)
M2
```

```
array([[ 1., -1.,  1.],
       [ 0.,  3.,  0.],
       [ 0.,  2.,  2.]])
```

```
M3 = scale(M2, 1, 1/3)
M3
```

```
array([[ 1., -1.,  1.],
       [ 0.,  1.,  0.],
       [ 0.,  2.,  2.]])
```

```
M4 = replace(M3, 2, 1, -2)
M4
```

```
array([[ 1., -1.,  1.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  2.]])
```

Clearly, the last two rows show that the system is inconsistent. Thus, $\vec{b}$ is not in col(A).

3. No, $null(A)$ is trivial. We can verify this directly:

$\vec{u} \in$ null(A) if and only if $A\vec{u} = \vec{0}$:

```
A = np.array([[1, -1], [2, 1], [-1, 3]])
A
```

```
array([[ 1, -1],
       [ 2,  1],
       [-1,  3]])
```

```
u = np.array([[1],[1]])
u
```

```
array([[1],
       [1]])
```

```
#computing Au

Au = A @ u
Au
```

```
array([[0],
       [3],
       [2]])
```

Thus, $\vec{u}$ is not in $null(A)$.

Example 3

Suppose A is an invertible $n \times n$ matrix. Then, by the Invertible Matrix Theorem (Theorem 4 in section 2.1), we have the following:

1. $col(A) = \mathbb{R}^n$
2. $null(A) = \{0\}$

Basis, Dimension, and Rank

Let $H \subset \mathbb{R}^n$ be a subspace. A **basis** for H is a linearly independent set in H that spans H.

According to the Invertible Matrix Theorem (Theorem 4 in section 2.1), the columns of an invertible $n \times n$ matrix form a linearly independent set that spans $\mathbb{R}^n$. Thus, the column space of any invertible matrix is a basis for $\mathbb{R}^n$. It's important to note that bases are not unique.

Example 4 (The Standard Basis)

Let

$$I_n = [\vec{e}_1 \quad \vec{e}_2 \quad ... \quad \vec{e}_n]$$

be the identity matrix. The columns of I_n, denoted by

$$\vec{e}_1, \vec{e}_2, ..., \vec{e}_n,$$

form a basis for $\mathbb{R}^n$, which is called the standard basis of $\mathbb{R}^n$. For instance, the standard basis elements for $\mathbb{R}^3$ are

$$\vec{e_1} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \vec{e_2} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \vec{e_3} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

The following cell computes the standard basis of $\mathbb{R}^4$

```
#the 4x4 identity matrix
I = np.eye(4)

print(' e1 = \n', np.array([I[0]]).T, '\n')
print(' e2 = \n', np.array([I[1]]).T, '\n')
print(' e3 = \n', np.array([I[2]]).T, '\n')
print(' e4 = \n', np.array([I[3]]).T, '\n')
```

```
e1 =
[[1.]
[0.]
[0.]
[0.]]

e2 =
[[0.]
[1.]
[0.]
[0.]]

e3 =
[[0.]
[0.]
[1.]
[0.]]

e4 =
[[0.]
[0.]
```

(continues on next page)

(continued from previous page)

```
[0.]
[1.]]
```

A basis for col A

The column space of a matrix A is generated by the set of all columns of A. If A is invertible, by the invertible matrix theorem, the columns are all pivot columns and form a linearly independent set and therefore, a basis for col A. In general, we have:

Theorem 13

The pivot columns of a matrix A form a basis for $col(A)$.

Example 5

Suppose

$$A = \begin{bmatrix} 1 & 3 & 3 & 2 & -9 \\ -2 & -2 & 2 & -8 & 2 \\ 2 & 3 & 0 & 7 & 1 \\ 3 & 4 & -1 & 11 & 8 \end{bmatrix}.$$

1. Find a basis for $col(A)$.
2. Find a basis for $null(A)$.

Solution

1. We reduce A to its reduced row echelon form (RREF) to find its pivot columns:

```
A = np.array([[1,3,3,2,-9], [-2,-2,2,-8,2], [2,3,0,7,1], [3,4,-1,11,8]])
A
```

```
array([[ 1,  3,  3,  2, -9],
       [-2, -2,  2, -8,  2],
       [ 2,  3,  0,  7,  1],
       [ 3,  4, -1, 11,  8]])
```

```
A1 = replace(A, 1, 0, 2)
A2 = replace(A1, 2, 0, -2)
A3 = replace(A2, 3, 0, -3)
A4 = scale(A3, 1, 1/4)
A5 = replace(A4, 2, 1, 3)
A6 = replace(A5, 3, 1, 5)
A7 = scale(A6, 2, 1/7)
A8 = replace(A7, 0, 1, -3)
A9 = replace(A8, 3, 2, -15)
A10 = replace(A9, 1, 2, 4)
A11 = replace(A10, 0, 2, -3)
A11
```

```
array([[ 1.,  0., -3.,  5.,  0.],
       [ 0.,  1.,  2., -1.,  0.],
       [ 0.,  0.,  0.,  0.,  1.],
       [ 0.,  0.,  0.,  0.,  0.]])
```

We can see from the RREF of A that A has 3 pivot columns, which are the first two columns and the last column. Therefore, a basis for col(A) is the set consisting of the first two columns and the last column of A:

$$\left\{ \begin{bmatrix} 1 \\ -2 \\ 2 \\ 3 \end{bmatrix}, \begin{bmatrix} 3 \\ -2 \\ 3 \\ 4 \end{bmatrix}, \begin{bmatrix} -9 \\ 2 \\ 1 \\ 8 \end{bmatrix} \right\}$$

It's important to note that the columns of the RREF of A are not necessarily in col(A), and we cannot use them to form a basis for col(A).

2. To find a basis for null(A), we solve $A\vec{x} = 0$ in the usual manner. Let's form the augmented matrix $[A \mid \vec{0}]$:

```
A0 = np.array([[1,3,3,2,-9,0], [-2,-2,2,-8,2,0], [2,3,0,7,1,0], [3,4,-1,11,8,0]])
A1 = replace(A0, 1, 0, 2)
A2 = replace(A1, 2, 0, -2)
A3 = replace(A2, 3, 0, -3)
A4 = scale(A3, 1, 1/4)
A5 = replace(A4, 2, 1, 3)
A6 = replace(A5, 3, 1, 5)
A7 = scale(A6, 2, 1/7)
A8 = replace(A7, 0, 1, -3)
A9 = replace(A8, 3, 2, -15)
A10 = replace(A9, 1, 2, 4)
A11 = replace(A10, 0, 2, -3)
A11
```

```
array([[ 1.,  0., -3.,  5.,  0.,  0.],
       [ 0.,  1.,  2., -1.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  1.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.]])
```

From this, we can see that the general solution is:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 3x_3 - 5x_4 \\ -2x_3 + x_4 \\ x_3 \\ x_4 \\ 0 \end{bmatrix} = x_3 \begin{bmatrix} 3 \\ -2 \\ 1 \\ 0 \\ 0 \end{bmatrix} + x_4 \begin{bmatrix} -5 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = span(\begin{bmatrix} 3 \\ -2 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -5 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix})$$

It is also clear that $\begin{bmatrix} 3 \\ -2 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} -5 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ are linearly independent. So they form a basis for $null(A)$.

Note that a subspace H can have different bases. For example, the following sets are both bases for $\mathbb{R}^2$:

$$B = \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\} \quad \text{and} \quad G = \left\{ \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix} \right\}$$

However, all bases have something in common:

Theorem 14

All bases of a subspace H in $\mathbb{R}^n$ have the same number of vectors.

The **dimension** of a subspace H, denoted as $dim(H)$, is the number of vectors in any of its bases. Furthermore, we define $dim(\{\vec{0}\})$ to be 0. The dimension of the column space of a matrix A, denoted as $rank(A)$, is referred to as the **rank** of A.

Example 6

Show that $dim(\mathbb{R}^n) = n$.

Solution:

We can use Example 6, where it was shown that the columns of the identity matrix I_n, denoted as $\{\vec{e}_1, \ldots, \vec{e}_n\}$, form a basis for $\mathbb{R}^n$.

Since there are n columns in the identity matrix I_n, the dimension of $\mathbb{R}^n$ is equal to n. Hence, $dim(\mathbb{R}^n) = n$.

Example 7

Find the rank of subspaces of $\mathbb{R}^3$.

Solution:

In $\mathbb{R}^3$, we have the following subspaces:

1. The zero subspace $\{\vec{0}\}$, which has zero dimension.
2. Lines passing through the origin, which have one dimension: every line in $\mathbb{R}^3$ can be spanned by a vector.
3. Planes passing through the origin, which have two dimensions: every plane in $\mathbb{R}^3$ can be spanned by two linearly independent vectors.
4. The entire space $\mathbb{R}^3$, which has three dimensions.

The next Theorem states that the sum of the dimensions of the column space and null space is equal to the number of columns in the matrix. Additionally, it provides a way to determine the dimensions of the column space and null space based on the number of pivot and non-pivot columns in the matrix, respectively.

Theorem 15 (Rank-Nullity Theorem)

If A be an $m \times n$ matrix, then $dim(col(A)) + dim(null(A)) = n$. Moreover,

1. $dim(col(A))$ = the number of pivot columns of A.
2. $dim(null(A))$ = the number of non-pivot columns of A.

Example 8

Find the rank and dimension of $null(A)$, where $A = \begin{bmatrix} 1 & 3 & 3 & 2 & -9 \\ -2 & -2 & 2 & -8 & 2 \\ 2 & 3 & 0 & 7 & 1 \\ 3 & 4 & -1 & 11 & 8 \end{bmatrix}$

Solution

From the solution of Example 7, We know that the RREF of A is

```
A11
```

```
array([[ 1.,  0., -3.,  5.,  0.,  0.],
       [ 0.,  1.,  2., -1.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  1.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.]])
```

Which has three pivot columns and two non-pivot columns. Therefore, $rank(A) = 3$ and $dim(null(A)) = 2$

Exercises

Exercises

1. Determine which one of the following sets are subspaces:

   a.

$$H = \left\{ \begin{bmatrix} x \\ y \\ z \end{bmatrix} \in \mathbb{R}^3 \text{ with } x \geq 0, \right\}.$$

   b.

$$H = \left\{ \begin{bmatrix} x \\ 1 \\ 2x \end{bmatrix} \in \mathbb{R}^3 \right\}.$$

2. Let

$$A = \begin{bmatrix} 1 & 2 & -9 \\ -2 & -2 & 2 \\ 2 & 3 & 0 \end{bmatrix}.$$

Find the dimension of $col(A)$ and $null(A)$.

3. Suppose A is a 5×5 matrix, and A has 3 pivot columns.

   a. Prove that $A\vec{x} = \vec{0}$ must have nontrivial solutions.

   b. Is there an example of A where $col(A) = null(A)$?

4. Find an explicit example of a 4×4 matrix A which has $colA = null(A)$

9.4.2 Introduction to Linear Transformations

This section introduces linear transformations (sometimes referred to as linear maps) and explores their relationship with linear systems. We discuss how the existence and uniqueness of solutions to linear systems can be reformulated in terms of linear transformations.

Motivation

In many applications, the matrix equation arises in a way that is not directly connected to linear combinations of vectors. In fact, we think of an $m \times n$ matrix A as a function that transforms $\vec{x} \in \mathbb{R}^m$ to $A\vec{x} \in \mathbb{R}^n$. With this perspective, finding the solution set of a linear system $A\vec{x} = \vec{b}$ is the same as finding the set of all $\vec{x}$ that A transforms into $\vec{b}$. Theorem 1 in the previous section states that:

1. $A(\vec{u} + \vec{v}) = A\vec{u} + A\vec{v}$
2. $A(c\vec{u}) = cA\vec{u}$

In general any function with these properties is called a *linear transformation*. More precisely, a function

$$T : \mathbb{R}^m \to \mathbb{R}^n$$

is a linear transformation if it satisfies the following conditions:

$$T(\vec{u} + \vec{v}) = T(\vec{u}) + T(\vec{v}) \quad \forall \vec{u}, \vec{v} \in \mathbb{R}^m$$

$$T(c\vec{u}) = cT(\vec{u}) \quad \forall \vec{u} \in \mathbb{R}^m \quad \text{and} \quad \forall c \in \mathbb{R}$$

These two conditions lead to the following useful fact:

$T : \mathbb{R}^m \to \mathbb{R}^n$ is a linear transformation if and only if

1. $T(\vec{0}) = \vec{0}$
2. $T(c\vec{u} + d\vec{v}) = cT(\vec{u}) + dT(\vec{v}) \quad \forall \vec{u}, \vec{v} \in \mathbb{R}^m \quad \text{and} \quad \forall c \in \mathbb{R}$

The last property can also be generalized to n vectors:

$$T(c_1\vec{u}_1 + c_2\vec{u}_2 + \cdots + c_p\vec{u}_p) = c_1T(\vec{u}_1) + c_2T(\vec{u}_2) + \cdots + c_pT(\vec{u}_p) \quad \text{where} \quad 1 \leq i \leq p$$

This property states that T preserves linear combinations of vectors.

Example 1

Show that

$T : \mathbb{R}^2 \to \mathbb{R}^3$ given by

$$T\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} 2x_2 \\ 0 \\ 3x_1 \end{bmatrix}.$$

is a linear transformation.

Solution:

T is linear as shown below:

1.

$$T\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}+\begin{bmatrix}y_1\\y_2\end{bmatrix}\right)=T\left(\begin{bmatrix}x_1+y_1\\x_2+y_2\end{bmatrix}\right)=\begin{bmatrix}2(x_2+y_2)\\0\\3(x_1+y_1)\end{bmatrix}=\begin{bmatrix}2x_2+2y_2\\0\\3x_1+3x_2\end{bmatrix}.$$

$$T\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}\right)+T\left(\begin{bmatrix}y_1\\y_2\end{bmatrix}\right)=\begin{bmatrix}2x_2\\0\\3x_1\end{bmatrix}+\begin{bmatrix}2y_2\\0\\3y_1\end{bmatrix}=\begin{bmatrix}2x_2+2y_2\\0\\3x_1+3x_2\end{bmatrix}.$$

2.

$$T\left(c\begin{bmatrix}x_1\\x_2\end{bmatrix}\right)=T\left(\begin{bmatrix}cx_1\\cx_2\end{bmatrix}\right)=\begin{bmatrix}2cx_2\\0\\3cx_1\end{bmatrix}.$$

$$cT\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}\right)=c\begin{bmatrix}2x_2\\0\\3x_1\end{bmatrix}=\begin{bmatrix}c2x_2\\0\\c3x_1\end{bmatrix}.$$

Let's write a Python function for T:

```
import numpy as np

def T(V):
    W = np.zeros((3, 1))   # return the zero vector in R^3
    W[0, 0] = 2 * V[1, 0]  # change the first component to 2*x_2
    W[2, 0] = 3 * V[0, 0]  # change the third component to 3*x_1
    return W
```

Now, let's plug in some vectors:

```
# input vectors
V = np.array([[1], [1]])

U = np.array([[0], [0]])

#output vectors
W1 = T(V)
W2 = T(U)

print('T(V): \n \n', W1)

print(10*'*')

print('\n T(U):  \n\n', W2)
```

```
T(V):

 [[2.]
 [0.]
 [3.]]
**********

 T(U):

 [[0.]
 [0.]
 [0.]]
```

Example 2

Determine if $T : \mathbb{R}^2 \to \mathbb{R}^3$ given by

$$T\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} 2x_2 \\ 1 \\ 3x_1 \end{bmatrix}.$$

is a linear transformation.

Solution: T is not linear, since

$$T\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \neq \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Example 3

Let

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Prove that the transformation

$$T : \mathbb{R}^3 \to \mathbb{R}^2$$

defined by $T(\vec{x}) = A\vec{x}$ is a linear transformation. T is called the **projection** onto the xy-plane.

Solution:

The transformation T is defined by a matrix multiplication. According to Theorem 1 in the previous section, the matrix product satisfies the linearity property. T is called a projection because when applied to

$$\vec{u} = \begin{bmatrix} x \\ y \\ z \end{bmatrix},$$

it doesn't affect the first two components and replaces z with zero. Let's verify this by evaluating it for

$$\vec{u} = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}:$$

```
# matrix A
A = np.array([[1,0,0],[0,1,0],[0,0,0]])

# vector U
U = np.array([[1,3,5]]).T


print('T maps \n\n', U,'\n\n to \n\n', A @ U)
```

```
T maps
```

(continues on next page)

(continued from previous page)

```
[[1]
[3]
[5]]

to

[[1]
[3]
[0]]
```

Example 4

Let

$$A = \begin{bmatrix} 1 & \lambda \\ 0 & 1 \end{bmatrix}.$$

Then, the transformation

$$T : \mathbb{R}^2 \to \mathbb{R}^2$$

defined by $T(\vec{x}) = A\vec{x}$ is a linear transformation and is called a **shear transformation**.

For $\lambda = 2$, we have:

```
# matrix A
A = np.array([[1,2],[0,1]])

# inpout data
e1 = np.array([1,0])
e2 = np.array([0,1])

#output
w1 = A @ e1
w2 = A @ e2


print('T maps \n\n', e1,'\n\n to \n\n', w1)

print(10*'*')

print('T maps \n\n', e2,'\n\n to \n\n', w2)
```

```
T maps

 [1 0]

 to

 [1 0]
**********
T maps
```

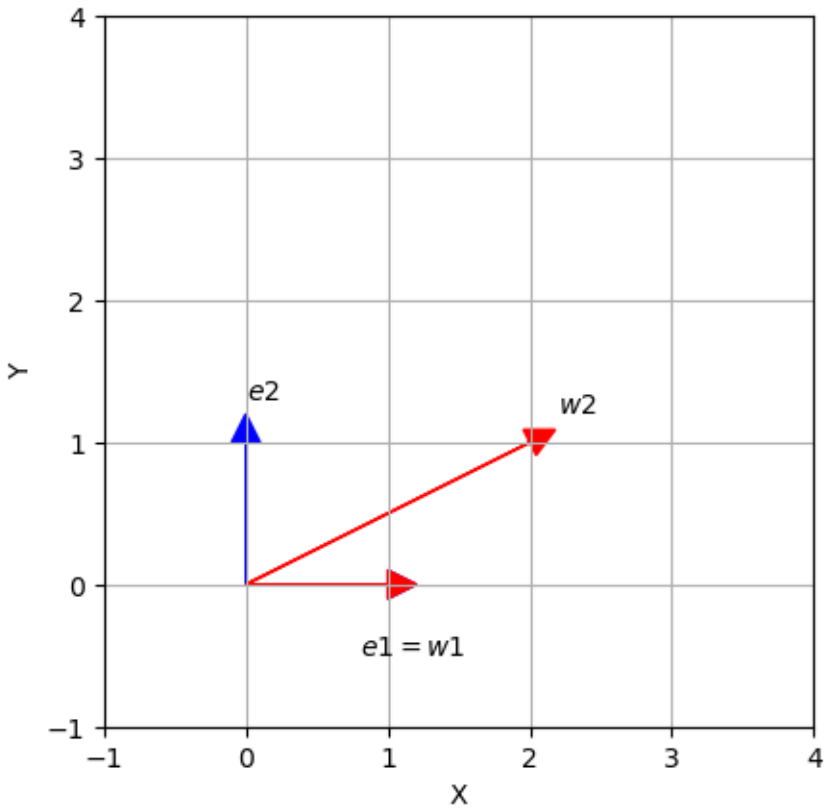
(continues on next page)

(continued from previous page)

```
[0 1]

to

[2 1]
```

Lets plot these vectors:

```
import matplotlib.pyplot as plt

ax = plt.axes()

ax.arrow(0, 0, 1, 0, head_width = 0.2,head_length = 0.2, fc ='b', ec ='b')
ax.arrow(0, 0, 1, 0, head_width = 0.2,head_length = 0.2, fc ='r', ec ='r')

ax.arrow(0, 0, 0, 1, head_width = 0.2,head_length = 0.2, fc ='b', ec ='b')
ax.arrow(0, 0, 2, 1, head_width = 0.2,head_length = 0.2, fc ='r', ec ='r')

ax.text(0.8,0 - 0.5,'$e1 = w1$')
ax.text(0,1.3,'$e2$')
ax.text(2.2,1.2,'$w2$')

ax.set_xticks(np.arange( -1, 5, step = 1))
ax.set_yticks(np.arange( -1, 5, step = 1))
ax.set_aspect('equal')

ax.set_xlabel("X")
ax.set_ylabel("Y")

plt.grid()
plt.show()
```



Observe that all points along the y-axis remain fixed while other points are shifted parallel to the y-axis by a distance proportional to their perpendicular distance from the x-axis; more specifically,

$$T(\begin{bmatrix} x \\ y \end{bmatrix}) = \begin{bmatrix} x + 2y \\ y \end{bmatrix}.$$

Shearing a plane figure does not change its area. The shear transformation can also be generalized to three dimensions:

Let

$$A = \begin{bmatrix} 1 & 0 & \lambda \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Then, the transformation

$$T : \mathbb{R}^3 \to \mathbb{R}^3$$

defined by $T(\vec{x}) = A\vec{x}$ is a shear transformation.

The standard matrix representation of a linear transformation

As we have seen previously, any $m \times n$ matrix A defines a linear transformation $T : \mathbb{R}^n \to \mathbb{R}^m$ given by $\vec{x} \mapsto A\vec{x}$. The converse of this statement is also true, as stated in the following theorem:

Theorem 16

Let $T : \mathbb{R}^n \to \mathbb{R}^m$ be a linear transformation. There exists a unique $m \times n$ matrix A such that $T(\vec{x}) = A\vec{x}$ for all $\vec{x} \in \mathbb{R}^n$. In fact, the ith column of matrix A is given by $T(\vec{e}_i)$, where $\vec{e}_i \in \mathbb{R}^n$ represents the ith column of the identity matrix I_n.

The matrix A is called **the standard matrix representation** of T. The name is motivated by the fact that we are using the standard basis of $\mathbb{R}^n$, $\vec{e}_i$s, to construct A.

Example 5

Find the standard matrix representation of the linear transformation in Example 1.

Solution:

We apply T to the standard basis vectors $\vec{e_1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\vec{e_2} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ to obtain the columns of the matrix representation D of T:

$$T(\vec{e_1}) = \begin{bmatrix} 2(0) \\ 0 \\ 3(1) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 3 \end{bmatrix}$$

$$T(\vec{e_2}) = \begin{bmatrix} 2(1) \\ 0 \\ 3(0) \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}$$

Thus, the standard matrix representation D of T is:

$$D = \begin{bmatrix} 0 & 2 \\ 0 & 0 \\ 3 & 0 \end{bmatrix}$$

Let's write Python code for this example and verify our solution:

```
def T(V):
    W = np.zeros((3,1)) #returns the zero vector in R^3
    W[0,0] = 2*V[1,0]   #changes the first component to 2*x_1
    W[2,0] = 3*V[0,0]   #changes the third component to 3*x_2
    return W

e1 = np.array([[1],[0]])
e2 = np.array([[0],[1]])

#the first column of A

c1 = T(e1)

#the second column of A

c2 = T(e2)

# formaing D

D = np.concatenate((c1,c2), axis =1)
```

(continues on next page)

(continued from previous page)

```
D
```

```
array([[0., 2.],
       [0., 0.],
       [3., 0.]])
```

In Example 1, we saw that $T(\begin{bmatrix}1\\1\end{bmatrix}) = \begin{bmatrix}2\\0\\3\end{bmatrix}$. Let's verify that $D\begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}2\\0\\3\end{bmatrix}$:

```
v = np.array([[1],[1]])

D @ v
```

```
array([[2.],
       [0.],
       [3.]])
```

Example 6

Consider the linear transformation $T : \mathbb{R}^2 \to \mathbb{R}^2$ defined as follows: $T(\vec{x})$ first reflects $\vec{x}$ across the x-axis and then rotates it 90° counterclockwise. Find the standard matrix representation of T.

Solution:

To find the standard matrix representation of T we apply T to the standard basis vectors of $\mathbb{R}^2$:

T reflects $\begin{bmatrix}1\\0\end{bmatrix}$ to itself and then rotates it to $\begin{bmatrix}0\\1\end{bmatrix}$.

T reflects $\begin{bmatrix}0\\1\end{bmatrix}$ to $\begin{bmatrix}0\\-1\end{bmatrix}$ and then rotates it to $\begin{bmatrix}1\\0\end{bmatrix}$.

Therefore, the standard matrix of T is

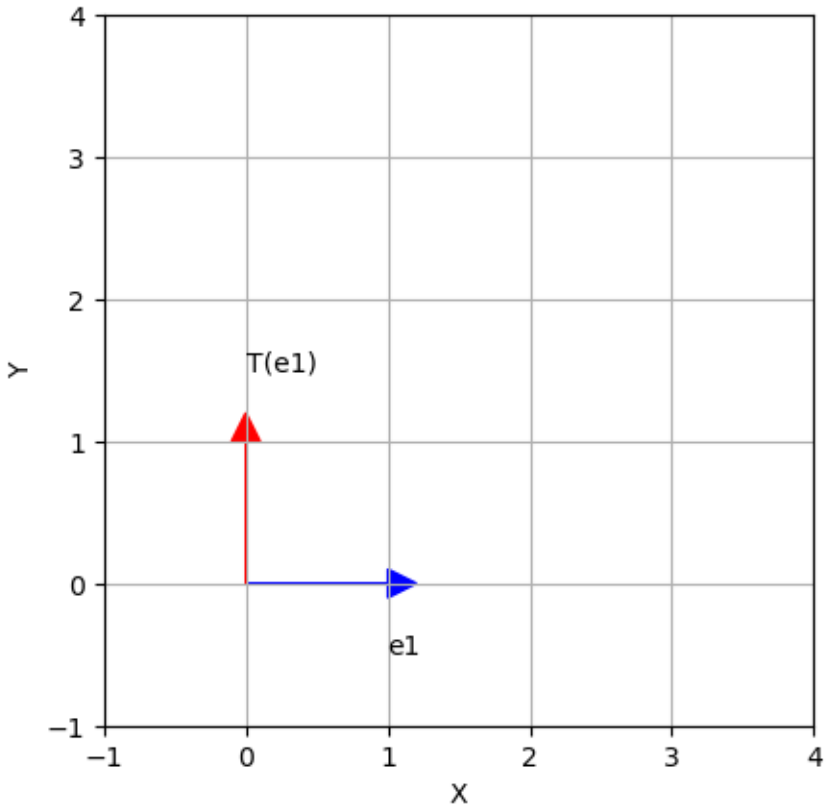
$$\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}.$$

```
import matplotlib.pyplot as plt

ax = plt.axes()

ax.arrow(0, 0, 1, 0, head_width = 0.2,head_length = 0.2, fc ='b', ec ='b')
ax.arrow(0, 0, 0, 1, head_width = 0.2,head_length = 0.2, fc ='r', ec ='r')


ax.text(1,0 - 0.5,'e1')
ax.text(0,1 + 0.5,'T(e1)')



ax.set_xticks(np.arange( -1, 5, step = 1))
ax.set_yticks(np.arange( -1, 5, step = 1))
ax.set_aspect('equal')
```

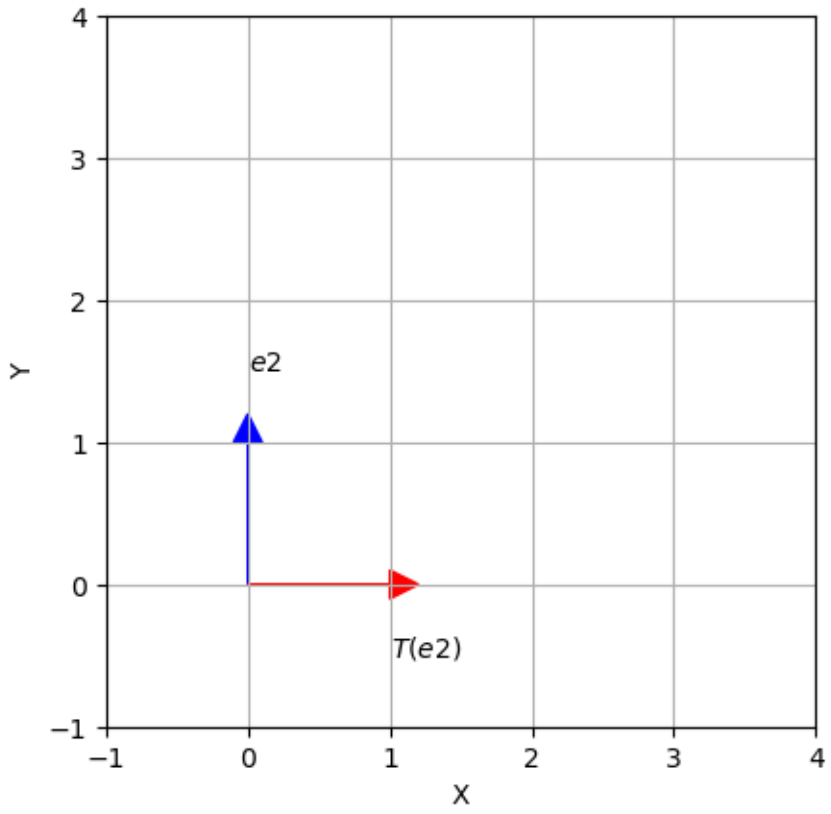
(continues on next page)

(continued from previous page)

```
ax.set_xlabel("X")
ax.set_ylabel("Y")

plt.grid()
plt.show()
```




```
ax = plt.axes()

ax.arrow(0, 0, 0, 1, head_width = 0.2,head_length = 0.2, fc ='b', ec ='b')
ax.arrow(0, 0, 1, 0, head_width = 0.2,head_length = 0.2, fc ='r', ec ='r')


ax.text(0,1 + 0.5,'$e2$')
ax.text(1,0 - 0.5,'$T(e2)$')


ax.set_xticks(np.arange( -1, 5, step = 1))
ax.set_yticks(np.arange( -1, 5, step = 1))
ax.set_aspect('equal')

ax.set_xlabel("X")
ax.set_ylabel("Y")

plt.grid()
plt.show()
```



Exercises

Exercises

1. Determine if each transformation is linear

   a. $T : \mathbb{R}^2 \to \mathbb{R}^2$ given by

$$T\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_2 \\ -x_1 \end{bmatrix}.$$

b. $T : \mathbb{R}^2 \to \mathbb{R}^2$ given by

$$T\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} (x_1)^2 \\ 0 \end{bmatrix}.$$

2. Find the standard matrix of $T : \mathbb{R}^2 \to \mathbb{R}^2$ given by $T(\vec{x}) =$ first reflect $\vec{x}$ across the x-axis and then rotate $45°$ counterclockwise.
3. Suppose $T : \mathbb{R}^2 \to \mathbb{R}^3$ is a linear map and

$$T\left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad \text{and} \quad T\left(\begin{bmatrix} 0 \\ 1 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}.$$

Find $T\begin{bmatrix} 2 \\ 3 \end{bmatrix}$.

9.5 Eigenvalues and Eigenvectors

```
import numpy as np
from numpy import linalg
```

9.5.1 Eigenvalues and Eigenvectors

This section introduces eigenvalues and eigenvectors of a square matrix and explores some of their applications. The goal of this section is to dissect the action of linear maps into elements that are easy to visualize.

9.5.2 Motivation

Consider a linear map $T : \mathbb{R}^n \to \mathbb{R}^m$ defined by $\vec{x} \mapsto A\vec{x}$. Although T moves $\vec{x}$ in a variety of directions, there are some vectors on which the action of T is easy to understand. For example, suppose $A = \begin{bmatrix} 3 & -2 \\ 1 & 0 \end{bmatrix}$, $\vec{u} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$. Let's compute $A\vec{u}$

```
A = np.array([[3,-2], [1,0]])
u = np.array([2,1])

A @ u
```

```
array([4, 2])
```

We can see that the image of $\vec{u} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ is

$$A\vec{u} = \begin{bmatrix} 4 \\ 2 \end{bmatrix} = 2 \begin{bmatrix} 2 \\ 1 \end{bmatrix} = 2\vec{u}$$

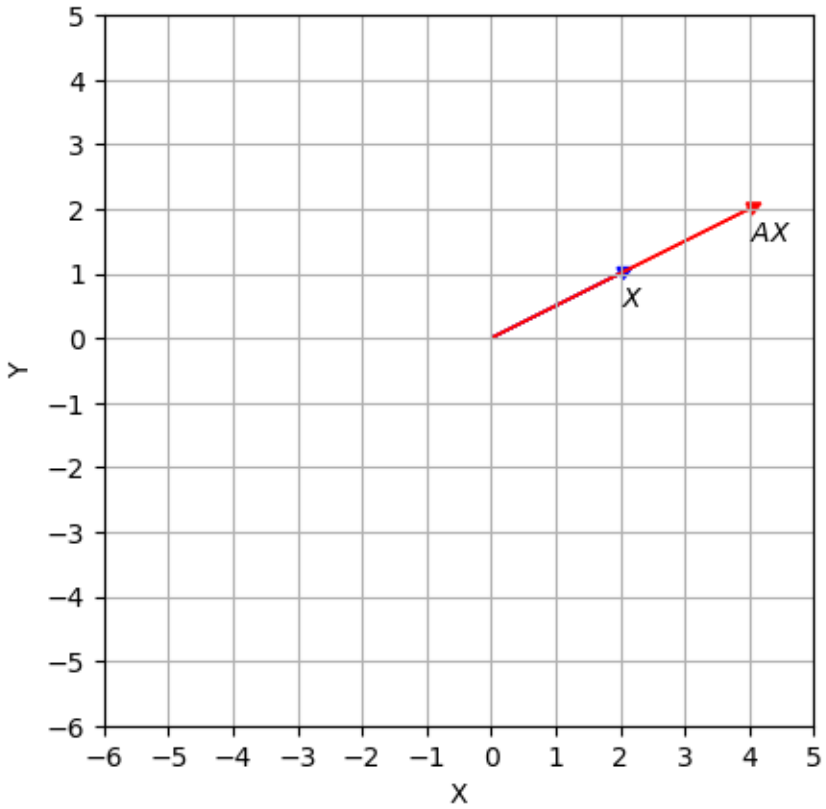
In fact, A *streches* $\vec{u}$.

```
import matplotlib.pyplot as plt


def plot_map(vector, matrix):
    x1 = vector[0]
    y1 = vector[1]

    x2 = (A @ vector)[0]
    y2 = (A @ vector)[1]

    ax = plt.axes()

    ax.arrow(0, 0, x1, y1, head_width = 0.2,head_length = 0.2, fc ='b', ec ='b')
    ax.arrow(0, 0, x2, y2, head_width = 0.2,head_length = 0.2, fc ='r', ec ='r')

    ax.text(x1,y1 - 0.5,'$X$')
    ax.text(x2,y2-0.5,'$AX$')

    z = max(np.abs(x1), np.abs(x2), np.abs(y1),np.abs(y2))

    ax.set_xticks(np.arange(-z-2, z+2, step = 1))
```

(continues on next page)

(continued from previous page)

```
    ax.set_yticks(np.arange(-z-2, z+2, step = 1))
    ax.set_aspect('equal')

    ax.set_xlabel("X")
    ax.set_ylabel("Y")

    plt.grid()
    plt.show()


plot_map(u, A)
```



We are interested in such special vectors that when transformed by matrix A, result in a scalar multiple of themselves. More generally, let A be an $n \times n$ matrix. We seek non-trivial solutions to equations of the form:

$$A\vec{x} = \lambda\vec{x} \quad \text{for some} \quad \lambda \in \mathbb{R}$$

If $\vec{u} \neq 0$ is a solution for some λ, we call λ an **eigenvalue** and $\vec{u}$ an **eigenvector** of A corresponding to λ.

Example 1

Consider the matrix equation:

$$A\vec{u} = \begin{bmatrix} 4 \\ 2 \end{bmatrix} = 2\begin{bmatrix} 2 \\ 1 \end{bmatrix} = 2\vec{u}$$

Observe that $\lambda = 2$ is an eigenvalue of A and $\vec{u}$ is the corresponding eigenvector.

a. Is $\vec{v} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ also an eigenvector of A?

b. Is $\lambda = 4$ an eigenvalue?

c. Show that $\lambda = 1$ is an eigenvalue and find an eigenvector for it.

Solution:

(a) Let's compute $A\vec{v}$:

```
v = np.array([1, -1])

A @ v
```

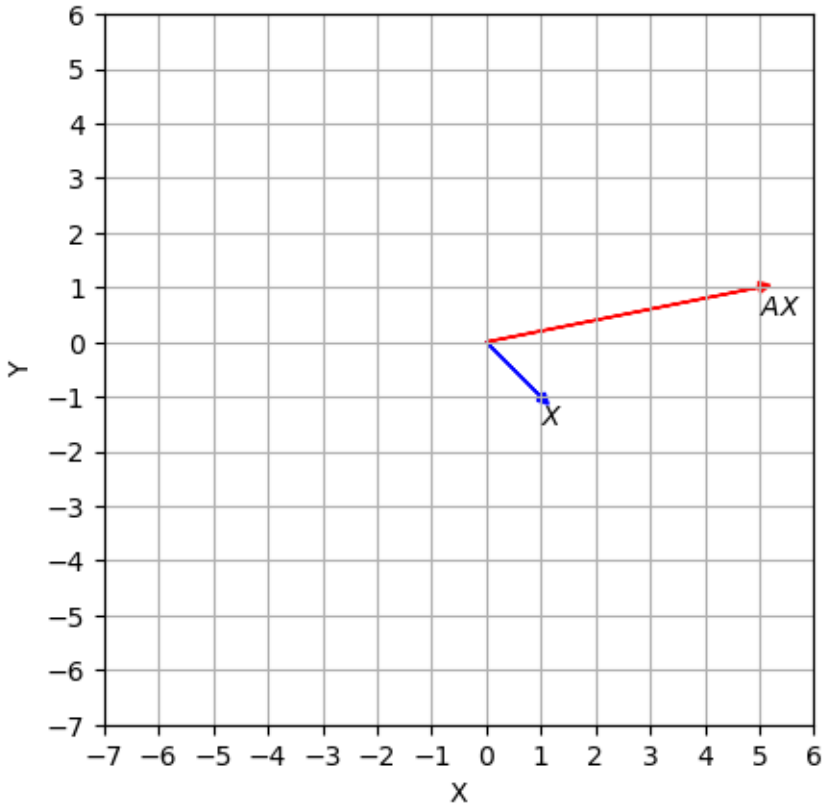
```
array([5, 1])
```

If there were a λ such that $A\vec{v} = \lambda\vec{v}$, we would have obtained:

$$\begin{bmatrix} 1 \\ -1 \end{bmatrix} = \lambda \begin{bmatrix} 5 \\ 1 \end{bmatrix}$$

Which is not possible for any value of λ, and as can be seen in the figure below, the vector $\vec{v}$ does not satisfy the eigenvalue equation.

```
plot_map(v, A)
```



(b) No. $\lambda = 4$ is a eigenvalue for A if and only if $A\vec{x} = 4\vec{x}$ has a non-trivial solution. Let's reformulate this problem in

terms of a homogenous equation which is easier to deal with:

$$A\vec{x} = 4\vec{x} \quad \iff \quad (A - 4I_2)\vec{x} = 0$$

The homogeneous equation $(A - 4I_2)\vec{x} = 0$ has a non-trivial solution if and only if it has a free variable (a non-pivot column).

```
# identity matrix I_2
I_2 = np.eye(2)

A - 4*I_2
```

```
array([[-1., -2.],
       [ 1., -4.]])
```

Clearly, the columns of $A - 4I$ are linearly independent; as a result, there is no free variable and the system has only the trivial solution $\vec{x} = 0$.

(c) $\lambda = 1$ is a eigenvalue for A if and only if $A\vec{x} = 1\vec{x}$ has a non-trivial solution.

$$A\vec{x} = \vec{x} \quad \iff \quad (A - I_2)\vec{x} = \vec{0}$$

```
A - I_2
```

```
array([[ 2., -2.],
       [ 1., -1.]])
```

Since the columns of $(A - I_2)$ are linearly independent, there is a free variable and $\lambda = 1$ is an eigenvalue of A. To find a corresponding eigenvector $\vec{u}$ we solve

$$(A - I_2)\vec{x} = 0$$

Let's set up the augmented matrix:

```
# the augmented vector
b = np.array([[0],[0]])

# add to the matrix
M = np.concatenate((A - I_2, b), axis = 1)
M
```

```
array([[ 2., -2.,  0.],
       [ 1., -1.,  0.]])
```

Let's recall the row operations:

```
def swap(matrix, row1, row2):

    copy_matrix=np.copy(matrix).astype('float64')

    copy_matrix[row1,:] = matrix[row2,:]
    copy_matrix[row2,:] = matrix[row1,:]

    return copy_matrix

# Replacing a row by the sum of itself and a multiple of another
```

(continues on next page)

(continued from previous page)

```
def replace(matrix, row1, row2, scalar):
    copy_matrix=np.copy(matrix).astype('float64')
    copy_matrix[row1] = matrix[row1]+ scalar * matrix[row2]
    return copy_matrix
```

```
M1 = swap(M, 0, 1)
M1
```

```
array([[ 1., -1.,  0.],
       [ 2., -2.,  0.]])
```

```
M2 = replace(M1, 1, 0, -2)
M2
```

```
array([[ 1., -1.,  0.],
       [ 0.,  0.,  0.]])
```

A general solution for $(A - I_2)\vec{x} = 0$ is

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ x_2 \end{bmatrix} = x_2 \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

and the set of all solutions is

$$\text{span}\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix} \right)$$

each non-zero element of this set is an eigenvector corresponding to $\lambda = 1$.

The above simple example leads to a general case:

Theorem 17

Let A be an $n \times n$ matrix and λ be a scalar. The following statements are equivalent:

1. λ is an eigenvalue of A.
2. $(A - \lambda I_n)\vec{x} = 0$ has non-trivial solution.
3. $det(A - \lambda I_n) = 0$

9.5.3 Eigenspace and characteristic polynomial

$null(A - \lambda I_n)$ is a subspace of $\mathbb{R}^n$ and contains the zero vector and all eigenvectors corresponding to λ. This subspace is known as the **eigenspace** of A corresponding to λ, and is denoted by E_λ. Intuitively, a matrix acts as dilations on its eigenspaces.

Example 2

The set

$$\text{span}\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix} \right)$$

Which we found in Example 1, is in fact the eigenspace E_1 of A corresponding to $\lambda = 1$. And any vector $\vec{u} \in E_1$ is preserved under the action of A (dilations factor = 1):

$$A\vec{u} = \vec{u}$$

The *polynomial* $P(\lambda) = \det(A - \lambda I_n)$ is called the **characteristic polynomial** of A. The zeros of this polynomial are precisely the eigenvalues of A. Thus, the polynomial can be factorized as:

$$P(\lambda) = (\lambda - \lambda_0)^{m_0}(\lambda - \lambda_1)^{m_1} \ldots (\lambda - \lambda_k)^{m_k}$$

For each λ_i, where $0 \leq i \leq k$, the power m_i is known as **the algebraic multiplicity** of λ_i.

Example 3

Suppose

$$A = \begin{bmatrix} 4 & -1 & 6 \\ 0 & 1 & 6 \\ 0 & 0 & 8 \end{bmatrix}.$$

We will find all eigenvalues and their multiplicities of A by finding its characteristic polynomial.

Solution:

The characteristic polynomial is given by:

$$\det(A - \lambda I_3) = \begin{vmatrix} 4-\lambda & -1 & 6 \\ 0 & 1-\lambda & 6 \\ 0 & 0 & 8-\lambda \end{vmatrix} = 0$$

Expanding the determinant, we obtain the characteristic polynomial:

$$P(\lambda) = (4 - \lambda)(1 - \lambda)(8 - \lambda),$$

The zeros of this polynomial (the eigenvalues of A) are $\lambda = 4$ with multiplicity 1, $\lambda = 1$ with multiplicity 1, and $\lambda = 8$ with multiplicity 1.

Example 4

Let A be the matrix given in Example 2. For each eigenvalue that we found in Example 2, find the corresponding eigenspace together with its dimension.

Solution:

The eigenspace corresponding to $\lambda = 4$:

$$E_4 = null(A - 4I) = \{\vec{x} : \quad (A - 4I)\vec{x} = 0\} =$$

$$\left\{ \begin{bmatrix} x \\ y \\ z \end{bmatrix} : \quad \begin{bmatrix} 0 & -1 & 6 \\ 0 & -3 & 6 \\ 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = 0 \right\} = \left\{ \begin{bmatrix} x \\ y \\ z \end{bmatrix} : \quad \begin{bmatrix} -y + 6z \\ -3y + 6z \\ 4z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right\}$$

Therefore,

$$E_4 = \left\{ \begin{bmatrix} x \\ 0 \\ 0 \end{bmatrix} : x \in \mathbb{R} \right\} = span\left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right) \quad \text{and} \quad dim(E_4) = 1$$

The eigenspace corresponding to $\lambda = 1$:

$$E_1 = null(A - I) = \{\vec{x} : \quad (A - I)\vec{x} = 0\} =$$

$$\left\{ \begin{bmatrix} x \\ y \\ z \end{bmatrix} : \quad \begin{bmatrix} 3 & -1 & 6 \\ 0 & 0 & 6 \\ 0 & 0 & 7 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = 0 \right\} = \left\{ \begin{bmatrix} x \\ y \\ z \end{bmatrix} : \quad \begin{bmatrix} 3x - y + 6z \\ 6z \\ 7z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right\}$$

Therefore,

$$E_1 = \left\{ \begin{bmatrix} x \\ 3x \\ 0 \end{bmatrix} : x \in \mathbb{R} \right\} = span\left(\begin{bmatrix} 1 \\ 3 \\ 0 \end{bmatrix} \right) \quad \text{and} \quad dim(E_1) = 1$$

The eigenspace corresponding to $\lambda = 8$

$$E_8 = null(A - 8I) = \{\vec{x} : \quad (A - 8I)\vec{x} = 0\} =$$

$$\left\{ \begin{bmatrix} x \\ y \\ z \end{bmatrix} : \quad \begin{bmatrix} -4 & -1 & 6 \\ 0 & -7 & 6 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = 0 \right\} = \left\{ \begin{bmatrix} x \\ y \\ z \end{bmatrix} : \quad \begin{bmatrix} -4x - y + 6z \\ -7y + 6z \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right\}$$

Therefore,

$$E_8 = \left\{ \begin{bmatrix} \frac{6}{4} \\ 1 \\ \frac{7}{6} \end{bmatrix} : y \in \mathbb{R} \right\} = span\left(\begin{bmatrix} 9 \\ 6 \\ 7 \end{bmatrix} \right) \quad \text{and} \quad dim(E_8) = 1$$

Before we move on to the next topic, let us check our calculation using `numpy.linalg.eig` from NumPy linear algebra:

```
A = np.array([[4, -1, 6], [0, 1, 6], [0, 0, 8]])

evalues , evectors = np.linalg.eig(A)

for i in range(evalues.shape[0]):
    print(evalues[i], ' is an eigenvector with ', evectors[:,i], ' as the␣
 ↪corresponding eigenvectors')
```

```
4.0  is an eigenvector with  [1. 0. 0.]  as the corresponding eigenvectors
1.0  is an eigenvector with  [0.31622777 0.9486833  0.        ]  as the␣
 ↪corresponding eigenvectors
8.0  is an eigenvector with  [0.69853547 0.46569032 0.54330537]  as the␣
 ↪corresponding eigenvectors
```

Note that `numpy.linalg.eig` returns the unit vector of eigenvalues, and that is why we see different eigen vectors. To check that let's normalize the eigenvector $\begin{bmatrix} 9 \\ 6 \\ 7 \end{bmatrix}$ corresponding to $\lambda = 8$:

```
v = np.array([9,6,7])

#normalizing v
e_v = v/np.linalg.norm(v)
e_v
```

```
array([0.69853547, 0.46569032, 0.54330537])
```

Let's have another look at Example 2. The eigenvalues are the entries on the main diagonal. This is always true for triangular matrices and can be proved by induction:

Theorem 3

Let A be a square triangular matrix. Then the eigenvalues of A are the entries on the diagonal.

Numerical Notes

1. Example 3 provides a method for manually computing eigenvectors in simple cases. In practical situations, using matrix program and row reduction method to find an eigenspace (for a specified eigenvalue) may not be entirely reliable. Roundoff error can occasionally lead to a reduced echelon form with the wrong number of pivots. The best computer programs compute approximations for eigenvalues and eigenvectors simultaneously, to any desired degree of accuracy, for matrices that are not too large.
2. Some software such as Mathematica and Maple can use symbolic calculations to find the characteristic polynomial of a moderate-sized matrix. But there is no formula or finite algorithm to solve the characteristic equation of a general matrix $n \times n$ for $n \geq 5$. The best numerical methods for finding eigenvalues avoid the characteristic polynomial entirely. In fact, MATLAB finds the characteristic polynomial of a matrix A by first computing the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ of A and then expanding the product

$$(\lambda - \lambda_1)(\lambda - \lambda_2) \ldots (\lambda - \lambda_n)$$

Exercises

Exercises

1. If $\vec{v}$ is an eigenvector of A corresponding to λ, what is $A^3\vec{v}$?
2. If A is an $n \times n$ matrix and 2 is an eigenvalue of A, show that 4 is an eigenvalue of $2A$.
3. If A is an invertible matrix with eigenvalue λ show that A^{-1} has eigenvalue $1/\lambda$.
4. Let

$$A = \begin{bmatrix} 1 & 3 & 3 \\ -3 & -5 & -3 \\ 3 & 3 & 1 \end{bmatrix}.$$

a. Find the eigenvalues of A and their multiplicities.

b. For each eigenvalue in part (a) find the corresponding eigenspace.

c. Find the dimension of each eigenspace in part (b) by finding a basis for it. How this result is related to your answer to part (a)?

9.5.4 Diagonalization

This section introduces diagonalization and characterizes diagonalizability. Diagonalization is a useful factorization of a matrix A based on its eigenvalue–eigenvector information. This factorization allows us to quickly access properties that are invariant under similarity, such as rank, invertibility, and eigenvalues. It also enables us to calculate powers of A for large k, which is a fundamental concept in various applications, including discrete dynamical systems.

Computing powers of a matrix

Let's consider the matrix $A = \begin{bmatrix} 7 & 2 \\ -4 & 1 \end{bmatrix}$. We wish to compute A^k for an arbitrary $k \in \mathbb{N}$, given that $A = PAP^{-1}$, where $P = \begin{bmatrix} 1 & 1 \\ -1 & -2 \end{bmatrix}$, and $D = \begin{bmatrix} 5 & 0 \\ 0 & 3 \end{bmatrix}$.

Using this decomposition, we have:

$$A^k = (PDP^{-1})(PDP^{-1})\dots(PDP^{-1}) = PD^kP.$$

Since D is diagonal, $D^k = \begin{bmatrix} 5^k & 0 \\ 0 & 3^k \end{bmatrix}$. Additionally, $P^{-1} = \begin{bmatrix} 2 & 1 \\ -1 & -1 \end{bmatrix}$. Therefore,

$$A^k = \begin{bmatrix} 1 & 1 \\ -1 & -2 \end{bmatrix} \begin{bmatrix} 5^k & 0 \\ 0 & 3^k \end{bmatrix} \begin{bmatrix} 2 & 1 \\ -1 & -1 \end{bmatrix} = \begin{bmatrix} 2 \times 5^k - 3^k & 5^k - 3^k \\ 2 \times 3^k - 2 \times 5^k & 2 \times 3^k - 5^k \end{bmatrix}.$$

This decomposition, known as diagonalization, enables us to compute A^k easily for large values of k. Without such factorization, computing A^k for large matrices and large k can be a time-consuming task. Therefore, it is beneficial to find such decomposition before calculating the power of A.

Unfortunately, this decomposition doesn't exist for all matrices. The goal of this section is to characterize matrices that admit such a factorization.

Diagonalization

An $n \times n$ matrix A is called **diagonalizable** if it is *similar* to a diagonal matrix. More precisely, there exists an invertible $n \times n$ matrix P and an $n \times n$ diagonal matrix D such that $A = PDP^{-1}$.

It turns out an $n \times n$ matrix is diagonalizable if we can form a basis $\mathbb{R}^n$ using eigenvectors of A:

Theorem 18

An $n \times n$ matrix A is diagonalizable if and only if A has n linearly independent eigenvectors.

In fact, suppose $\lambda_1, \lambda_2, \dots, \lambda_n$ are eigenvalues of A, and $\{\vec{v}_1, \vec{v}_2, \dots, \vec{v}_n\}$ is the linearly independent set of eigenvectors of A. If we form the matrix P by taking the eigenvectors as columns, and D is a diagonal matrix with eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$ on the diagonal, then we have $A = PDP^{-1}$.

Finding eigenvectors in general is not an easy task, and is not clear how to check which $n \times n$ matrices have n linearly independent eigenvectors. The next theorem partially addresses this issue.

Theorem 19

The eigenvectors corresponding to distinct eigenvalues are linearly independent.

Combining Theorem 18 and Theorem 19, we obtain the following corollary:

Corollary 1

An $n \times n$ matrix with n distinct eigenvalues is diagonalizable.

Example 1

Is

$$A = \begin{bmatrix} 1 & 3 & 2 \\ -3 & -5 & 3 \\ 3 & 3 & 1 \end{bmatrix}$$

diagonalizable? If yes find a diagonalization for it.

Solution

To determine if A is diagonalizable we need to find eigenvalues and to check if it admits 3 linearly independent eigenvectors.

- Step 1

we use *linalg.eigvas* to find the eigenvalues and eigenvectors of A:

```
A = np.array([[1,3,3],[-3,-5,3],[3,3,1]])
# find evalues and evectors
evalues, evectors = linalg.eig(A)

print("eigenvalues = ",  evalues)
```

```
eigenvalues =  [-5. -2.  4.]
```

A has three distinct eigenvalues and therefore it is diagonalizable by Corollary 1.

- Step 2: Constructing the diagonal matrix D

```
lambda_1 = evalues[0]
lambda_2 = evalues[1]
lambda_3 = evalues[2]

D = np.array([[lambda_1, 0, 0], [0, lambda_2, 0], [0, 0, lambda_3]])
D
```

```
array([[-5.,  0.,  0.],
       [ 0., -2.,  0.],
       [ 0.,  0.,  4.]])
```

- Step 3 : Finding the matrix P whose ith column is an eigenvector corresponding to λ_i for $i \in \{1,\ 2,\ 3\}$:

```
# set up the invirtible matrix P
P = evectors
```

- Step 4: compute the inverse of matrix P

```
# compute the inverse of P

Q = linalg.inv(P)
```

- Step 5 (sanity check): Now that we have D, P, $Q = P^{-1}$, lets check if $A = PDQ$

```
P @ D @ Q
```

```
array([[ 1.,  3.,  3.],
       [-3., -5.,  3.],
       [ 3.,  3.,  1.]])
```

Example 2

Is

$$A = \begin{bmatrix} 1 & 3 & 2 \\ -3 & -5 & -3 \\ 3 & 3 & 1 \end{bmatrix}$$

diagonalizable? If yes find a diagonalization for it.

Solution:

- Step 1: Finding eigenvalues and eigenvectors of A:

```
A = np.array([[1,3,3],[-3,-5,-3],[3,3,1]])

# find evalues and evectors
evalues, evectors = linalg.eig(A)

print("eigenvalues = ",  evalues)
```

```
eigenvalues =  [ 1. -2. -2.]
```

A has 2 eigenvalues: $\lambda_1 = 1$ with multiplicity 1 and $\lambda_2 = -2$ with multiplicity 2. Note that we cannot use Corollary 1 anymore because eigenvalues are not distinct. To check if A is diagonalizable, we need to find three linearly independent eigenvectors of A. Lets have a look at our eigenvectors:

```
# A matrix whose columns are eigenvectors:
P = evectors

# computes the rank of P
np.linalg.matrix_rank(P)
```

```
3
```

Since $rank(P) = 3$ the columns of P are linearly independents. Thus, A is diagonalizable by Theorem 1.

- Step 2: Finding the diagonal matrix D

```
lambda_1 = evalues[0]
lambda_2 = evalues[1]
lambda_3 = evalues[2]
```

(continues on next page)

(continued from previous page)

```
D = np.array([[lambda_1, 0, 0], [0, lambda_2, 0], [0, 0, lambda_3]])
D
```

```
array([[ 1.,  0.,  0.],
       [ 0., -2.,  0.],
       [ 0.,  0., -2.]])
```

- Step 3: Finding the matrix P whose ith column is an eigenvector corresponding to λ_i for $i \in \{1, 2, 3\}$. We already have this matrix so we go to the next step.
- Step 4: Computing the inverse of P

```
# computing the inverse of P

Q = np.linalg.inv(P)
Q
```

```
array([[-1.73205081, -1.73205081, -1.73205081],
       [ 0.89919543,  2.75763333,  1.8584379 ],
       [-0.6541932 ,  0.5254512 ,  1.1796444 ]])
```

- Step 5 (sanity check): lets check if $A = PDQ$

```
# check A = PDQ
P @ D @ Q
```

```
array([[ 1.,  3.,  3.],
       [-3., -5., -3.],
       [ 3.,  3.,  1.]])
```

The next theorem provides another way to characterize the diagonalizability of matrices based on the multiplicity of eigenvalues. Before stating the theorem, we need the following definition:

Suppose A is a square matrix and λ is an eigenvalue of A. The dimension of the corresponding eigenspace $E_\lambda = null(A - \lambda I)$ is called the **geometric multiplicity of** λ.

Theorem 20

A square matrix A is diagonalizable if and only if the algebraic multiplicity of each eigenvalue is equal to its geometric multiplicity. More formally, if $\lambda_1, \lambda_2, \dots, \lambda_p$ are eigenvalues of A with algebraic multiplicities $m_1, m_2, \dots, m_p$, respectively, then

$$m_i = \dim(E_{\lambda_i}) \quad \forall i \in \{1, 2, \dots, p\}$$

Recall that $\dim(E_{\lambda_i})$ = dim null$(A - \lambda_i I)$ = the number of non-pivot columns in $(A - \lambda_i I)$

Example 3

Is $M = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 2 \end{bmatrix}$

diagonalizable? If so, find a diagonalization for it.

Solution:

M is an upper triangular matrix, and its eigenvalues are on the diagonal: $\lambda_1 = 1$ with algebraic multiplicity 2 and $\lambda_2 = 2$ with algebraic multiplicity 1.

Since we don't have distinct eigenvalues, Corollary 1 doesn't apply here. Let's take a look at our eigenvectors:

```
import numpy as np
from numpy import linalg


M = np.array([[1, 1, 0], [0, 1, 1], [0, 0, 2]])

evalues, evectors  = np.linalg.eig(M)
p = evectors

np.linalg.matrix_rank(p)
```

```
2
```

Observe that the rank of matrix P is two, which means we don't have three linearly independent eigenvectors in M. However, using Theorem 3, we can confirm that this is not the case, and consequently, M is not diagonalizable. In fact, it is straightforward to check that the geometric multiplicity of $\lambda = 1$ is 1 ($\dim(E_1) = 1$), while the algebraic multiplicity is 2. Thus, M is not diagonalizable.

Diagonalization as a change of basis

Recall that similar matrices represent the same linear map under two different choices of a pair of bases for $\mathbb{R}^n$ and $\mathbb{R}^m$. More precisely, suppose $A = PCP^{-1}$ (C is not necessarily diagonal), then the matrix representation of T with respect to the basis B (formed by the columns of P) is C. Conversely, changing the basis of $\mathbb{R}^n$ leads to a matrix representation of T that is similar to A (section 3.4). This particularly applies to diagonalizable matrices because a matrix is diagonalizable if it is similar to a diagonal matrix.

Theorem 20 (Diagonal Matrix Representation)

Suppose $T : \mathbb{R}^n \to \mathbb{R}^n$ and A is the standard matrix representation of T that is diagonalizable. Let $A = PDP^{-1}$ be a diagonalization of A where D is an $n \times n$ diagonal matrix, and P is an invertible $n \times n$ matrix. If B is the basis for $\mathbb{R}^n$ formed by the columns of P, then D is the matrix representation of the transformation $T : (\mathbb{R}^n, B) \to (\mathbb{R}^n, B)$.

Example 3 Let $T : \mathbb{R}^2 \to \mathbb{R}^2$ and

$$A = \begin{bmatrix} 7 & 2 \\ -4 & 2 \end{bmatrix}$$

be the standard matrix representation of T. Find a basis B for T such that the matrix representation of T with respect to B is diagonal.

Solution:

From Example 1, we know that $A = PAP^{-1}$ where $P = \begin{bmatrix} 1 & 1 \\ -1 & -2 \end{bmatrix}$ and $D = \begin{bmatrix} 5 & 0 \\ 0 & 3 \end{bmatrix}$.

The columns of P are eigenvectors of A, and they form a basis B for $\mathbb{R}^2$. By Theorem 2, D is the matrix representation of T with respect to B. In fact, the mappings $\vec{x} \to A\vec{x}$ and $\vec{x} \to D\vec{x}$ describe the same linear transformation, relative to different bases.

Exercises

1. Suppose

$$M = \begin{bmatrix} 2 & 4 & 3 \\ -4 & -6 & -3 \\ 3 & 3 & 1 \end{bmatrix}.$$

a. Is M diagonalizable? If yes find a diagonalization for it.

b. Use this diagonalization to compute M^{100}

2. Suppose

$$C = \begin{bmatrix} 3 & 0 & 1 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}.$$

Is C diagonalizable?

3. Let A be a 5×5 matrix, with two eigenvalues. One eigenspace is three-dimensional, and the other is two-dimensional. Is A diagonalizable?

4. Compute A^{200} where

$$A = \begin{bmatrix} 4 & -3 \\ 2 & -1 \end{bmatrix}.$$

9.5.5 Discrete Dynamical Systems

In this section, we will explore the application of diagonalization to discrete dynamical systems, focusing on their long-term behavior.

Modeling Dynamical Systems as sequence of linear systems

In many fields (e.g., ecology, biology, economics, etc.), we wish to mathematically model and study a dynamic system that changes over time. We are usually given several features of the system that are measured at (discrete) different time intervals which can be viewed as a sequence of vectors

$$\vec{x_0}, \vec{x_1}, \vec{x_2}, ...$$

We interpret $\vec{x_i}$ as the state of a system at time $t = i$. We also assume that there is a matrix A such that

$$\vec{x_n} = A\vec{x_{n-1}} \quad \forall n = 1, 2, 3, ... \quad (*)$$

Our goal is to develop a formula for x_n and describe what can happen when n approaches to infinity.

Let's consider the movement of populations or groups of people from one region to another. We want a simple model that considers the changes in the population of a city and its surrounding suburbs over a period of years. Let's say the initial year is 2020 and denote the population of the city and suburbs by c_0 and s_0, respectively. Now we can form the initial

vector population $\vec{x_0} = \begin{bmatrix} c_0 \\ r_0 \end{bmatrix}$. Similarly, for 2021 and subsequent years, denote the populations of the city and suburbs by the vectors

$$\vec{x_1} = \begin{bmatrix} c_1 \\ s_1 \end{bmatrix}, \ \vec{x_2} = \begin{bmatrix} c_2 \\ s_2 \end{bmatrix}, \ \vec{x_3} = \begin{bmatrix} c_3 \\ s_3 \end{bmatrix}, \ ...$$

How can be these vectors mathematically related?

Suppose demographic studies show that each year about 5% of the city's population moves to the suburbs (and 95% remains in the city), while 3% of the suburban population moves to the city (and 97% remains in the suburbs). With this information, we can find the vector population $\vec{x_1}$ based on the initial population $\vec{x_0}$. After the first year, $c_0 95\%$ remain in the city and $c_0 0.05$ move to the suburbs. Additionally, $s_0 .97\%$ remain in the suburbs and $s_0 0.03$ move to the city. Therefore,

$$\begin{bmatrix} c_1 \\ s_1 \end{bmatrix} = c_0 \begin{bmatrix} 0.95 \\ 0.05 \end{bmatrix} + s_0 \begin{bmatrix} 0.03 \\ 0.97 \end{bmatrix} = \begin{bmatrix} 0.95 & 0.03 \\ 0.97 & 0.05 \end{bmatrix} \begin{bmatrix} c_0 \\ s_0 \end{bmatrix}.$$

The matrix $M = \begin{bmatrix} 0.95 & 0.03 \\ 0.97 & 0.05 \end{bmatrix}$ is called the **migration matrix**. Consequently, we can write

$$\vec{x_n} = M\vec{x_{n-1}} \quad \forall n = 1, 2, 3, ...$$

and the sequence

$$\vec{x_0}, \vec{x_1}, \vec{x_2}, ...$$

describes the population of the city/suburbs over a period of years.

Example 1

Compute the population of the region just described for the years 2021 and 2022, given that the initial population in 2020 was $600,000$ in the city and $40,000$ in the suburbs.

Solution:

```
import numpy as np

# migration matrix
M = np.array([[0.95, 0.03] ,[0.05, 0.097]])


# initial population vector x_0
x_0 = np.array([600000, 40000])

# population vector x_1
x_1 = M @ x_0

print('the population of city in 2021 was = ', x_1[0],'\n')
print('the population of suburbs in 2021 was = ', x_1[1], '\n\n')

# population vector x_2

x_2 = M @ x_1

print('the population of the city in 2022 was = ', x_2[0],'\n')
print('the population of suburbs in 2022 was = ', x_1[1], '\n\n')
```

```
the population of city in 2021 was =  571200.0

the population of suburbs in 2021 was =  33880.0


the population of the city in 2022 was =  543656.4

the population of suburbs in 2022 was =  33880.0
```

Long-term behavior of dynamical systems

Eigenvalues and eigenvectors can be used to understand the long-term behavior or evolution of a dynamical system described by an equation in the form of $\vec{x}_{k+1} = A\vec{x}_k$. We assume that A is diagonalizable with n linearly independent eigenvectors $\{\vec{v_1}, \vec{v}_2, \dots, \vec{v}_n\}$ and corresponding eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$. For convenience, let's assume the eigenvectors are arranged such that $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$. Since $\{\vec{v_1}, \vec{v}_2, \dots, \vec{v}_n\}$ forms a basis for $\mathbb{R}^n$, any initial vector $\vec{x}_0$ can be expressed uniquely as:

$$\vec{x}_0 = c_1\vec{v_1} + c_2\vec{v_2} + \cdots + c_n\vec{v_n}$$

By applying A to both sides, we have:

$$\vec{x}_1 = A\vec{x}_0 = c_1\lambda_1\vec{v_1} + c_2\lambda_2\vec{v_2} + \cdots + c_n\lambda_n\vec{v_n}$$

And in general:

$$\vec{x}_k = A\vec{x}_{k-1} = c_1(\lambda_1)^k\vec{v_1} + c_2(\lambda_2)^k\vec{v_2} + \cdots + c_n(\lambda_n)^k\vec{v_n} \quad (**)$$

The above equation represents the evolution of the system over time. It shows that each component of the vector $\vec{x}_k$ is a linear combination of the eigenvectors $\vec{v}_i$, scaled by the corresponding eigenvalues λ_i raised to the power of k.

As an example, let's consider a predator-prey system involving owls and wood rats. The population vectors $\vec{x}_k$ at time k are given by:

$$\vec{x}_k = \begin{bmatrix} O_k \\ R_k \end{bmatrix}$$

where O_k is the population of owls and R_k is the population of rats (measured in thousands). The populations evolve according to the following equations:

$$O_{k+1} = (0.5)O_k + (0.4)R_k$$
$$R_{k+1} = -pO_k + (1.1)R_k$$

Here, the coefficient 0.5 in the first equation says that, without wood rats for food, only half of the owls will survive each month. The coefficient 1.1 in the second equation says that, in the absence of owls as predators, the rat population will grow by 10% each month. The parameter p is a positive value to be specified.

To determine the evolution of this system, we can compute the eigenvalues and eigenvectors of the coefficient matrix A, and then apply equation $(**)$. For $p = 0.104$, lets find the evolution of this system:

```
# coefficient matrix
A = np.array([[0.5, 0.4],[-0.104, 1.1]])

# Computing the eigenvalues and eigenvectors of A
np.linalg.eigvals(A)
```

```
array([0.58, 1.02])
```

Note that $|\lambda_1| \leq 1$ and $|\lambda_2| \geq 1$. A straightforward calculation shows that the corresponding eigenvectors are $\vec{v_1} = \begin{bmatrix} 5 \\ 1 \end{bmatrix}$ and $\vec{v_2} = \begin{bmatrix} 10 \\ 13 \end{bmatrix}$, respectively. The initial vector $\vec{x_0}$ can be expressed as a linear combination of the eigenvectors:

$$\vec{x_0} = d_1\vec{v_1} + d_2\vec{v_2} = [\vec{v_1}, \vec{v_2}] \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}$$

Therefore, according to equation $(**)$, we have:

$$\vec{x_k} = d_1(\lambda_1)^k\vec{v_1} + d_2(\lambda_2)^k\vec{v_2}$$

As $k \to \infty$, the term $(\lambda_1)^k\vec{v_1}$ approaches zero, allowing us to approximate:

$$x_k \approx d_2(\lambda_2)^k\vec{v_2}$$

Furthermore, observe that:

$$x_{k+1} \approx d_2(\lambda_2)^{k+1}\vec{v_2} = \lambda_2\vec{x_k} = 1.02\vec{x_k}$$

The above approximation shows that both the number of owls and rats grow by an approximate factor of 1.02 each month. let's calculate some population vectors, assuming the initial population vector is $\begin{bmatrix} 10 \\ 20 \end{bmatrix}$.

```
import numpy as np

# A matrix whose columns are eigenvectors
evectors = np.array([[5, 10], [1, 13]])

# Initial vector
x_0 = np.array([1, 2])

# Solving the equation for d_1 and d_2
d = np.linalg.solve(evectors, x_0)

# Extracting d_2
d_2 = d[1]

# Eigenvalues corresponding to the eigenvectors
evalues = np.array([10, 20])  # Replace with the actual eigenvalues

# Iterating over the range
for k in range(10):
    # Calculating the population using the eigenvalues, eigenvectors, and d_2
    population = d_2 * pow(evalues[1], k+1) * evectors[1]
    print('At month', k+1, 'the owl population is', population[0],' and ', 'the rat␣
↪population is', population[1],'\n')
```

```
At month 1 the owl population is 3.2727272727272725  and  the rat population is 42.
↪54545454545454

At month 2 the owl population is 65.45454545454545  and  the rat population is 850.
↪9090909090909
```

(continues on next page)

(continued from previous page)

```
At month 3 the owl population is 1309.090909090909  and  the rat population is␣
↪17018.181818181816

At month 4 the owl population is 26181.81818181818  and  the rat population is␣
↪340363.63636363635

At month 5 the owl population is 523636.36363636365  and  the rat population is␣
↪6807272.7272727275

At month 6 the owl population is 10472727.272727273  and  the rat population is␣
↪136145454.54545456

At month 7 the owl population is 209454545.45454547  and  the rat population is␣
↪2722909090.909091

At month 8 the owl population is -27786072.436363637  and  the rat population is -
↪361218941.6727273

At month 9 the owl population is 147091381.52727273  and  the rat population is␣
↪1912187959.8545456

At month 10 the owl population is 130576309.52727273  and  the rat population is␣
↪1697492023.8545456
```

Note that if the absolute values of both eigenvalues are less than 1, then both populations will tend towards zero for large values of k. If $\lambda_1 = 1$ and $\lambda 2 < 1$, then

$$x_k \approx d_1(\lambda_1)\vec{v_1}$$

Finally, if $\lambda_1 = 1$ and $\lambda 2 \geq 1$, then

$$x_k = d_1(\lambda_1)\vec{v_1} + d_2(\lambda_2)^k\vec{v_2}$$

Exercise

Exercises

1. Let A be a 2×2 matrix with eigenvalues 5 and $\frac{1}{5}$ and corresponding eigenvectors

$$\begin{bmatrix}1\\1\end{bmatrix} and \begin{bmatrix}-1\\1\end{bmatrix}.$$

Let $\vec{x}_k$ be a solution of $\vec{x}_{k+1} = A\vec{x}_k$, $\vec{x}_0 = \begin{bmatrix}9\\1\end{bmatrix}$.

a. Compute $\vec{x}_1 = A\vec{x}_0$

b. Find a formula for $\vec{x}_k$ involving k and the eigenvectors of A.

c. Describe what happens when $k \to \infty$.

9.6 Orthogonality

```
import numpy as np
from matplotlib import pyplot as plt
```

9.6.1 Inner Products and Orthogonality

In this section, we explore inner products and orthogonality. Inner products allow us to define intuitive geometric concepts such as length, distance, and perpendicularity in n dimensional space $\mathbb{R}^n$.

Inner product

Let $\vec{u}$ and $\vec{v}$ be vectors in $\mathbb{R}^n$. We can view them as $n \times 1$ matrices. The transpose of $\vec{u}$, denoted as $\vec{u}^T$, is a $1 \times n$ matrix. The matrix product of $\vec{u}^T$ and $\vec{v}$ yields a 1×1 matrix, which can be written as a real number (a scalar) without brackets. This scalar is known as the inner product or dot product of $\vec{u}$ and $\vec{v}$. To compute it, we multiply the corresponding elements of $\vec{u}$ and $\vec{v}$ and sum the results. More precisely, if

$$\vec{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}$$

and

$$\vec{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix},$$

then the dot product $\vec{u} \cdot \vec{v}$ is given by:

$$\vec{u} \cdot \vec{v} = \vec{u}^T \vec{v} = \begin{bmatrix} u_1 & u_2 & \dots & u_n \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = u_1 v_1 + u_2 v_2 + \cdots + u_n v_n$$

Example 1

Compute $\vec{u} \cdot \vec{v}$ for $\vec{u} = \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix}$ and $\vec{v} = \begin{bmatrix} -2 \\ 3 \\ 4 \end{bmatrix}$.

Solution

We use `numpy.dot()` to compute the dot product in Python:

```
import numpy as np

u = np.array([1, -1, 2])
v = np.array([-2, 3, 4])

uv = np.dot(u,v)
uv
```

```
3
```

Theorem 1 (Properties of the dot product)

Let $\vec{u}$, $\vec{v}$ and $\vec{w}$ be vectors in $\mathbb{R}^n$, and $c \in \mathbb{R}$ be a scalar. Then

1. $\vec{u} \cdot \vec{v} = \vec{v} \cdot \vec{u}$.
2. $(\vec{u} + \vec{v}) \cdot \vec{w} = \vec{u} \cdot \vec{w} + \vec{v} \cdot \vec{w}$.
3. $c(\vec{u} \cdot \vec{v}) = (c\vec{u}) \cdot \vec{v} = \vec{u} \cdot (c\vec{v})$.
4. $\vec{u} \cdot \vec{v} \geq 0$, and $\vec{u} \cdot \vec{u} = 0$ if and only if $\vec{u} = 0$.

combining (2) and (3), and using induction we can show:

$$(c_1\vec{u_1} + c_2\vec{u}_2 \ldots c_p\vec{u}_p) \cdot \vec{w} = c_1\vec{w} \cdot \vec{u}_1 + c_2\vec{w} \cdot \vec{u}_2 + \ldots c_p\vec{w} \cdot \vec{u}_p$$

Length of vectors

The dot product can be used to define the length of vectors: let $\vec{u} \in \mathbb{R}^n$ then the magnitude or the *length* of $\vec{u}$, denoted by $\|\vec{u}\|$ is defined by

$$\|\vec{u}\| = \sqrt{\vec{u} \cdot \vec{u}}.$$

- Note that this definition of length coincides with the standard notion of length in $\mathbb{R}^2$ and $\mathbb{R}^3$.
- A vector with a length of 1 is called a unit vector. For any vector $\vec{u}$, there exists a unit vector in the direction of $\vec{u}$. To obtain this vector, we first calculate the length of $\vec{u}$ and then divide $\vec{u}$ by its length $\|\vec{u}\|$. The resulting vector is referred to as the unit vector of $\vec{u}$, and commonly denoted as $\vec{e}_u$:

$$\vec{e}_u = \frac{\vec{u}}{\|\vec{u}\|}$$

This process is often called normalizing a vector, as it transforms $\vec{u}$ into a unit vector by scaling it to have a length of 1.

Example 2

Let $\vec{u} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$. Compute the following:

1. $\|\vec{u}\|$
2. $\| -2\vec{u}\|$
3. $\vec{e}_u$

Solution:

```
# (1)

u = np.array([1,2,3])

uu = np.dot(u,u)

length_u = np.sqrt(uu)
```

(continues on next page)

(continued from previous page)

```
length_u
```

```
3.7416573867739413
```

```
# (2)

v = -2*u

vv = np.dot(v,v)

length_v = np.sqrt(vv)

length_v
```

```
7.483314773547883
```

We could also use the properties of the dot products to compute $\| -2\vec{u}\|$:

$$\| -2\vec{u}\| = \sqrt{-2\vec{u} \cdot -2\vec{u}} = 2(\vec{u} \cdot \vec{u}) = 2\|\vec{u}\|$$

```
#(3) computing the unit vector of u

e_u = u/length_u

e_u
```

```
array([0.26726124, 0.53452248, 0.80178373])
```

Let's check that the length of $\vec{e}_u$ is exactly 1:

```
length_e = np.sqrt(np.dot(e_u,e_u))

length_e
```

```
1.0
```

Example 3

Given vectors

$$\vec{u} = \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix}$$

and

$$\vec{v} = \begin{bmatrix} -2 \\ 3 \\ 4 \end{bmatrix},$$

and the subspace W spanned by $\vec{u}$ and $\vec{v}$, find unit vectors $\vec{w}_1$ and $\vec{w}_2$ that form a basis for W.

Solution:

Since $\vec{u}$ and $\vec{v}$ are linearly independent (they are not scalar multiples of each other), we can proceed to normalize them to get unit vectors.

```
# unit vector of u
u = np.array([1,-1,2])
e_u = u / np.sqrt(np.dot(u,u))


# unit vector of v
v = np.array([-2, 3, 4])
e_v = v / np.sqrt(np.dot(v,v))


print("e_u = ", e_u)
print("e_v = ", e_v)
```

```
e_u =  [ 0.40824829 -0.40824829  0.81649658]
e_v =  [-0.37139068  0.55708601  0.74278135]
```

Distance in $\mathbb{R}^n$

For $\vec{u}$ and $\vec{v}$ in $\mathbb{R}^n$, the distance between $\vec{u}$ and $\vec{v}$, denoted by dist $(\vec{v}, \vec{u})$, is the length of their difference vector $\vec{u} - \vec{v}$. That is,

$$\text{dist}\,(\vec{u}, \vec{v}) = \|\vec{u} - \vec{v}\|.$$

Note that this definition coincides with the standard definition of distances in $\mathbb{R}^2$ and $\mathbb{R}^3$.

Example 4

Compute the distance between vectors in Example 2.

Solution:

```
w = u - v

dist = np.sqrt(np.dot(w,w))

print("The distance between ", u, " and ", v, " is ", dist)
```

```
The distance between  [1 2 3]  and  [-2 -4 -6]  is  11.224972160321824
```

Angles and Orthogonality

Theorem 2 (Cauchy-Schwarz inequality)

For $\vec{u}$ and $\vec{v}$ in $\mathbb{R}^n$ we have

$$|\vec{u} \cdot \vec{v}| \leq \|\vec{u}\|\|\vec{v}\|.$$

The Cauchy-Schwarz inequality gives us a way to define the notion of an angle between two n dimensional vectors $\vec{u}$ and $\vec{v}$. This notion also coincides with our intuition in $\mathbb{R}^2$ and $\mathbb{R}^3$:

Suppose $\vec{u}, \vec{v} \in \mathbb{R}^n$ are non-zero vectors. Note that

$$-1 \leq \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\|\|\vec{v}\|} \leq 1$$

Therefore, there is a unique $\theta \in [0, \pi]$ such that

$$\cos(\theta) = \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\|\|\vec{v}\|}$$

θ is referred to as the angle between $\vec{u}$ and $\vec{v}$ and is equal to

$$\theta = \arccos(\frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\|\|\vec{v}\|})$$

Example 5

Find the angle between vectors in Example 2.

Solution:

We first compute the length of $\vec{u}$ and $\vec{v}$ and then substitute them in above formula:

```
u_len = np.sqrt(np.dot(u,u))
v_len = np.sqrt(np.dot(v,v))

z = u_len * v_len

t = np.arccos(np.dot(u,v)/z)
print("The angle between u and v is ", t ," rad")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[6], line 1
----> 1 u_len = np.sqrt(np.dot(u,u))
      2 v_len = np.sqrt(np.dot(v,v))
      4 z = u_len * v_len

NameError: name 'u' is not defined
```

Two vectors $\vec{u}$ and $\vec{v}$ are **orthogonal**, denoted by $\vec{u} \perp \vec{v}$, if the angle θ between them is $\frac{\pi}{2}$. Alternatively, this condition is satisfied if and only if $\vec{u} \cdot \vec{v} = 0$. It is worth noting that the zero vector $\vec{0}$ is orthogonal to every vector because $\vec{0}^T \cdot \vec{u} = 0$ for any vector $\vec{u}$.

Observe that for any two vectors $\vec{u}$ and $\vec{v}$ we have

$$\|\vec{u} + \vec{v}\|^2 = (\vec{u} + \vec{v}) \cdot (\vec{u} + \vec{v}) = \|\vec{u}\|^2 + \|\vec{v}\|^2 + 2\vec{u} \cdot \vec{v}.$$

The above calculation implies the famous Pythagorean Theorem:

Theorem 3 (The Pythagorean Theorem)

Two vectors $\vec{u}$ and $\vec{v}$ are orthogonal if and only if

$$\|\vec{u} + \vec{v}\|^2 = \|\vec{u}\|^2 + \|\vec{v}\|^2$$

The orthogonal complement of a subspace

Let $W \subset \mathbb{R}^n$ be a subspace. If a vector $\vec{u}$ is orthogonal to every vector in the subspace W, then $\vec{u}$ is said to be orthogonal to W. The set of all vectors $\vec{u}$ that are orthogonal to W form a subspace which is commonly called **the orthogonal complement of** W and is denoted by $W^\perp$ (or simply $W^\perp$):

$$W^\perp = \{\vec{u} \in \mathbb{R}^n : \vec{u} \cdot \vec{w} = 0 \text{ for all } \vec{w} \in W\}$$

Example 6

Let W be the xy-plane in $\mathbb{R}^3$, and let L be the z-axis. Our intuition says L is orthogonal to W. In fact, if $\vec{z}$ and $\vec{w}$ are nonzero vectors such that $\vec{z} \in L$ and $\vec{w} \in W$, then $\vec{z}$ is orthogonal to $\vec{w}$. Since $\vec{w}$ and $\vec{z}$ were chosen arbitrarily, every vector on L is orthogonal to every $\vec{w}$ in W. In fact, L consists of all vectors that are orthogonal to the $\vec{w}$'s in W, and W consists of all vectors $\vec{w}$ orthogonal to the $\vec{z}$'s in L. Mathematically,

$$L = W^\perp \quad \text{and} \quad W = L^\perp$$

Example 7

If $W = span\left\{\begin{bmatrix}1\\0\\0\end{bmatrix}\right\}$, find a basis for $W^\perp$.

Solution:

Note that W is simply the x-axis in $\mathbb{R}^3$. Intuitively, $W^\perp$ should be the yz-plane, and a basis for it is

$$B = \left\{\begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}\right\}.$$

Alternatively, we can find the solutions to the linear system:

$$\begin{bmatrix}1 & 0 & 0\end{bmatrix}\begin{bmatrix}x\\y\\z\end{bmatrix} = \vec{0}.$$

We have:

$$\begin{bmatrix}x\\y\\z\end{bmatrix} = \begin{bmatrix}0\\y\\z\end{bmatrix} = y\begin{bmatrix}0\\1\\0\end{bmatrix} + z\begin{bmatrix}0\\0\\1\end{bmatrix}.$$

We conclude this section with a theorem summarizing the properties of the orthogonal complement of a subspace.

Theorem 4

Let W be a subspace of $\mathbb{R}^n$.

1. The orthogonal complement $W^\perp$ is also a subspace of $\mathbb{R}^n$.
2. The sum of the dimensions of W and $W^\perp$ is equal to the dimension of the ambient space $\mathbb{R}^n$, i.e., $dim(W) + dim(W^\perp) = n$.
3. If W is the column space of some matrix A, i.e., $W = \text{col}(A)$, then $W^\perp$ is precisely the null space of the transpose of A, i.e., $W^\perp = \text{null}(A^T)$.

Exercises

Exercises

1. Let $\vec{a} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$.

   a. Find a vector $\vec{w}$ that is in the opposite direction of $\vec{a}$ and has a magnitude of 2.

   b. Find two non-parallel vectors $\vec{u}$ and $\vec{v}$ which are both orthogonal to $\vec{a}$

2. Find two non-parallel vectors $\vec{u}$ and $\vec{v}$ which are both orthogonal to $\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$ and $\begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix}$.

3. Let

$$\vec{v} = \begin{bmatrix} 1 \\ 2 \\ 7 \end{bmatrix}$$

and

$$\vec{w} = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}.$$

a. Find $\vec{z} = \vec{v} - \left(\frac{\vec{v} \cdot \vec{w}}{\vec{w} \cdot \vec{w}} \right) \vec{w}$.

b. Check that $\vec{z}$ is orthogonal to $\vec{w}$.

4. If

$$W = span \left\{ \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \right\}$$

find a basis for $W^{\perp}$.

9.6.2 Orthogonal Projection

In this section, we explore the construction of orthogonal projections, which is a key step in many calculations involving orthogonality, and properties of projection maps.

Orthogonal Set

A set of vectors $\{\vec{u}_1, \vec{u}_2, \dots, \vec{u}_p\}$ in $\mathbb{R}^n$ is called an *orthogonal set* if each pair of distinct vectors from the set is orthogonal, meaning that $\vec{u}_i \cdot \vec{u}_j = 0$ whenever $i \neq j$. It is important to note that an orthogonal set, consisting of **nonzero vectors**, is always a linearly independent set. For that reason, if a basis for a subspace W of $\mathbb{R}^n$ happens to be an orthogonal set, it is called an **orthogonal basis**.

In general, an orthogonal basis makes computations easier. The following theorem demonstrates how:

Theorem 5

Let

$$B = \{\vec{u}_1, \vec{u}_2, \dots, \vec{u}_p\}$$

be an orthogonal basis for a subspace $W \subset \mathbb{R}^n$. Moreover, suppose $y \in W$ can be expressed as a linear combination of the basis elements:

$$\vec{y} = c_1\vec{u}_1 + c_2\vec{u}_2 + \cdots + c_p\vec{u}_p.$$

Then the weights c_i are given by

$$c_i = \frac{\vec{u}_i \cdot \vec{y}}{\vec{u}_i \cdot \vec{u}_i} \quad \forall i \in \{1, 2, \dots, p\}.$$

Example 1

Write the vector

$$\vec{y} = \begin{bmatrix} 6 \\ 1 \\ -8 \end{bmatrix}$$

as a linear combination of the vectors in

$$S = \left\{ \begin{bmatrix} 3 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 2 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ -4 \\ 7 \end{bmatrix} \right\}.$$

Solution:

Since S is an orthogonal set with three nonzero elements, it forms a basis for $\mathbb{R}^3$. Therefore, we can express $\vec{y}$ in terms of the vectors in S:

$$\vec{y} = c_1 \begin{bmatrix} 3 \\ 1 \\ 1 \end{bmatrix} + c_2 \begin{bmatrix} -1 \\ 2 \\ 1 \end{bmatrix} + c_3 \begin{bmatrix} -1 \\ -4 \\ 7 \end{bmatrix}.$$

We can solve this vector equation for c_is using techniques we learned in section 1.2. However, Theorem 1 suggests an alternative way of computing the coefficients:

$$c_i = \frac{\vec{u}_i \cdot \vec{y}}{\vec{u}_i \cdot \vec{u}_i} \quad \text{for all } i \in \{1, 2, 3\}.$$

By substituting the appropriate values of $\vec{u}_i$ and $\vec{y}$ into the formula, we can compute the coefficients c_is.

```
y = np.array([6, 1, -8])

u1 = np.array([3, 1, 1])

u2 = np.array([-1, 2, 1])

u3 = np.array([-1,-4, 7])

#computing the weights:

c1 = np.dot(y,u1)/ np.dot(u1,u1)

c2 = np.dot(y,u2)/ np.dot(u2,u2)

c3 = np.dot(y,u3)/ np.dot(u3,u3)

print("c1 = ", c1, ", c2 = ", c1, ", c3 = ", c3)
```

```
c1 =  1.0 , c2 =  1.0 , c3 =  -1.0
```

Orthogonal Projections onto 1-dimensional spaces

Given a nonzero vector $\vec{y}$ in $\mathbb{R}^n$, decomposing $\vec{y}$ into a sum of two vectors is a straightforward task. For example, for any choice of $\alpha, \beta \in \mathbb{R}$ such that $\alpha + \beta = 1$, we have:

$$\vec{y} = \alpha\vec{y} + \beta\vec{y}.$$

A more challenging question is when we want to express $\vec{y}$ as the sum of two vectors, where one is a scalar multiple of a given vector $\vec{u}$ and the other is orthogonal to $\vec{u}$. More specifically, we seek to write:

$$\vec{y} = \vec{p} + \vec{z},$$

where $\vec{p} = \alpha\vec{u}$ for some scalar α, and $\vec{z} \perp \vec{p}$. In other words, we are looking for a scalar α such that $\vec{z} := \vec{y} - \alpha\vec{u}$ is orthogonal to $\vec{u}$, which can be expressed as:

$$(\vec{y} - \alpha\vec{u}) \cdot \vec{u} = 0.$$

Simplifying the equation above, we find:

$$\alpha = \frac{\vec{y} \cdot \vec{u}}{\vec{u} \cdot \vec{u}}, \quad \text{and thus} \quad \vec{p} = \frac{\vec{y} \cdot \vec{u}}{\vec{u} \cdot \vec{u}}\,\vec{u}, \quad \text{and} \quad \vec{z} = \vec{y} - \vec{p}.$$

The vector $\vec{p}$, sometimes denoted as $\text{proj}_{\vec{u}}(\vec{y})$, is called the **orthogonal projection of $\vec{y}$ onto $\vec{u}$**, and $\vec{z}$ is referred to as the **component of $\vec{y}$ orthogonal to $\vec{u}$.**

Example 2

Suppose

$\vec{y} = \begin{bmatrix} 7 \\ 6 \end{bmatrix}$ and $\vec{u} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$.

Compute the $\text{proj}_{\vec{u}}(\vec{y})$ and the component of $\vec{y}$ orthogonal to $\vec{u}$

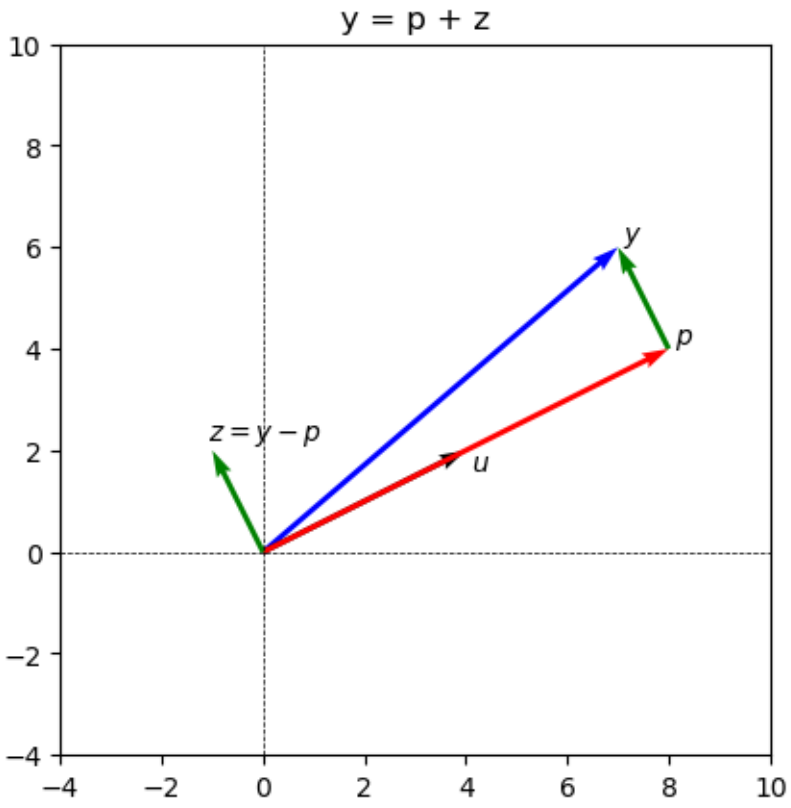
Solution:

```
# setup vectors

y = np.array([7,6])

u = np.array([4,2])

# projection of y onto u

proj_y_u = (np.dot(y,u)/ np.dot(u,u))* u
print( "the projection of y onto u is ", proj_y_u)

z = y - proj_y_u

print("the component of y orthogonal to u is", z )
```

```
the projection of y onto u is  [8. 4.]
the component of y orthogonal to u is [-1.  2.]
```

Let's plot $\text{proj}_{\vec{u}}(\vec{y})$ and the component of $\vec{y}$ orthogonal to $\vec{u}$ in xy-plane:

```
# Plot the coordinates in two separate plots
fig, ax = plt.subplots()

figsize=(20, 10)

# vector y
ax.quiver(0, 0, 7, 6, angles='xy', scale_units='xy', scale=1, color='blue', label ='y
 ↪')
ax.text(7.1,6.1,'$y$')

# vector u
ax.quiver(0, 0, 4, 2, angles='xy', scale_units='xy', scale=1, color='black')
ax.text(4.1,1.6,'$u$')

# orthogonal projection p
ax.quiver(0, 0, 8, 4, angles='xy', scale_units='xy', scale=1, color='red')
ax.text(8.1,4.1,'$p$')

# vector z
ax.quiver(0, 0, -1, 2, angles='xy', scale_units='xy', scale=1, color='green')
ax.text(-1.1,2.2,'$z = y - p$')

# A copy of vector z starting at the end of vector p
ax.quiver(8, 4, -1, 2, angles='xy', scale_units='xy', scale=1, color='green')



ax.set_xlim([-4, 10])
ax.set_ylim([-4, 10])
ax.set_aspect('equal')
ax.axhline(y=0, color='black', linestyle='--', linewidth=0.5)
ax.axvline(x=0, color='black', linestyle='--', linewidth=0.5)

ax.set_title('y = p + z')


plt.show()
```



Orthogonal Projections onto a general subspace

Let's take another look at the above Figure from Example 2. The projection of $\vec{y}$ onto any vector in the direction of $\vec{u}$, results in the same vector $\text{proj}_{\vec{u}}(\vec{y})$. In fact, what matters to us is the line that contains $\vec{u}$ (span$(\{\vec{u}\})$). With this in mind, we can interpret $\text{proj}_{\vec{u}}(\vec{y})$ as the projection of $\vec{y}$ onto a subspace of $\mathbb{R}^n$ generated by $\vec{u}$, and we can write it as $\text{proj}_{\text{span}(L)}(\vec{y})$. In fact, the notion of projection can be generalized to any subspace of $\mathbb{R}^n$.

Theorem 6

Let W be a subspace of $\mathbb{R}^n$. Then each $\vec{y} \in \mathbb{R}^n$ can be written uniquely as

$$\vec{y} = \vec{p} + \vec{z}$$

where $\vec{p}$ is in W and $\vec{z}$ is in $W^{\perp}$. Moreover, if $B = \{\vec{u}_1, \vec{u}_2, \ldots \vec{u}_p\}$ is any orthogonal basis of W, then

$$\vec{p} = \frac{\vec{y} \cdot \vec{u}_1}{\vec{u}_1 \cdot \vec{u}_1} + \frac{\vec{y} \cdot \vec{u}_2}{\vec{u}_2 \cdot \vec{u}_2} + \cdots + \frac{\vec{y} \cdot \vec{u}_p}{\vec{u}_p \cdot \vec{u}_p} \quad (**) \quad \text{and} \quad \vec{z} = \vec{y} - \vec{p}$$

The vector $\vec{p}$, sometimes denoted by $\text{proj}_W(\vec{y})$ is called the **orthogonal projection** of $\vec{y}$ onto W, and $\vec{z}$ is called the component of $\vec{y}$ orthogonal to W.

Example 3

Suppose

$$\vec{u}_1 = \begin{bmatrix} 2 \\ 5 \\ -1 \end{bmatrix},$$

$$\vec{u}_2 = \begin{bmatrix} -2 \\ 1 \\ 1 \end{bmatrix},$$

$W = \text{span}(\vec{u}_1, \vec{u}_2)$ and

$$\vec{y} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}.$$

Compute the $\text{proj}_W(\vec{y})$ and the component of $\vec{y}$ orthogonal to $\vec{u}$.

Solution:

$\vec{u}_1$ and $\vec{u}_2$ are orthogonal and form a basis for W; thus, we can use Theorem 3 to compute the projection of $\vec{y}$ onto W:

```
# setup vectors

u1 = np.array([2,5,-1])

u2 = np.array([-2,1,1])

y = np.array([1,2,3])

# projection of y onto W

proj_y_W = (np.dot(y,u1)/ np.dot(u1,u1))* u1  +  (np.dot(y,u2)/ np.dot(u2,u2))* u2

print( "The projection of y onto W is ", proj_y_W, '\n')

# component of y orthogonal to p
z = y - proj_y_W

print("The component of y orthogonal to u is", z )
```

```
The projection of y onto W is  [-0.4  2.   0.2]

The component of y orthogonal to u is [1.4 0.  2.8]
```

Note that if $\vec{y}$ happens to be in W then $\text{proj}_W(\vec{y}) = \vec{y}$. We can verify this fact for $\vec{u}_1$ which we know belongs to W:

```
proj_u_W = (np.dot(u1,u1)/ np.dot(u1,u1))* u1  +  (np.dot(u1,u2)/ np.dot(u2,u2))* u2
print(proj_u_W)
```

```
[ 2.  5. -1.]
```

The Best Approximation Theorem

Projections can be viewed as linear transformations:

Consider a subspace W of $\mathbb{R}^n$, and let $T_W : \mathbb{R}^n \to W$ be the mapping that sends a vector $\vec{y}$ to its orthogonal projection $\text{proj}_W(\vec{y})$. It is known that T_W is a linear transformation. If P represents the matrix of T_W, then $P^2 = P$. More precisely,

$$P\vec{w} \in W \quad \forall \vec{w} \in W$$

Informally, we say P does not move elements of W. The following theorem states that if $\vec{u}$ is not in W, then P sends it to *the closest point* in W from $\vec{u}$.

Theorem 7 (The Best Approximation Theorem)

Let W be a subspace of $\mathbb{R}^2$, and let $\vec{y} \in \mathbb{R}^n$. The projection of $\vec{y}$ onto W, denoted as $\vec{p} = \text{proj}_W(\vec{y})$, is the closest point in W to $\vec{y}$ in the following sense:

$$\|\vec{y} - \vec{p}\| \leq \|\vec{y} - \vec{w}\| \quad \forall \vec{w} \in W$$

In particular, if $\vec{y} \in W$, then the closest point is exactly $\vec{y}$. We refer to $\vec{p}$ as the best approximation to $\vec{y}$ by elements of W. The real number $\|\vec{y} - \vec{p}\|$) is interpreted as the distance between the vector $\vec{y}$ and the subspace W.

Example 4

Suppose

$$\vec{u}_1 = \begin{bmatrix} 5 \\ -2 \\ 1 \end{bmatrix},$$

$$\vec{u}_2 = \begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix},$$

$W = \text{span}(\vec{u}_1, \vec{u}_2)$

and

$$\vec{y} = \begin{bmatrix} -1 \\ -5 \\ 10 \end{bmatrix}.$$

1. What is the closest point to $\vec{y}$ in W?
2. Find the distance between $\vec{y}$ and W.

Solution

1. The closest point is the projection of $\vec{y}$ onto W:

```
# setup vectors

y = np.array([-1, -5, 10])
```

(continues on next page)

(continued from previous page)

```
u1 = np.array([5, -2, 1])

u2 = np.array([1 , 2, -1])

# projection of y onto W

p = (np.dot(y,u1)/ np.dot(u1,u1))* u1  +  (np.dot(y,u2)/ np.dot(u2,u2))* u2

print( "The projection of y onto W is ", p, '\n')


# component of y orthogonal to p
z = y - p

print("The component of y orthogonal to u is", z, '\n' )

# distance between y and W
r = np.sqrt(np.dot(z,z))
print("The distance between y and W is", r )
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[9], line 3
      1 # setup vectors
----> 3 y = np.array([-1, -5, 10])
      5 u1 = np.array([5, -2, 1])
      7 u2 = np.array([1 , 2, -1])

NameError: name 'np' is not defined
```

It is possible to simplify the $(**)$ formula in Theorem 6 which computes projection. First, we need the following definition:

A basis B for a subspace W is called an **orthonormal basis** if B is an orthogonal basis and every element of B is a unit vector. Any orthogonal basis can be turned into an orthonormal basis by normalizing every basis element.

Theorem 8

Let W be a subspace of $\mathbb{R}^n$ and $B = \{\vec{u}_1, \vec{u}_2, \ldots \vec{u}_p\}$ be an orthonormal basis of W. Then, for every $\vec{y} \in \mathbb{R}^n$, the projection of $\vec{y}$ onto W is

$$\text{proj}_W(\vec{y}) = \vec{y} \cdot \vec{u_1} + \vec{y} \cdot \vec{u_2} + \cdots + \vec{y} \cdot \vec{u_p}$$

We can also represent this formula even in a more compact form:

if $U = \left[\vec{u}_1, \vec{u_2}, \ldots, \vec{u}_p\right]$ then

$$\text{proj}_W(\vec{y}) = UU^T\vec{y} \quad (*)$$

A matrix whose columns are orthonormal, such as U, is called an orthogonal matrix. For an $n \times n$ orthogonal matrix U, besides (*), we have

$$U^TU = I_n$$

Exercises

Exercises

1. Let

$$B = \left\{ \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right\}$$

be a basis of $\mathbb{R}^2$.

a. Check that this is an orthogonal basis.

b. Find the coordinates of the vector

$$\begin{bmatrix} 3 \\ 5 \end{bmatrix}$$

with respect to the basis B

2. Suppose

$$\vec{u}_1 = \begin{bmatrix} -7 \\ 1 \\ 4 \end{bmatrix},$$

$$\vec{u}_2 = \begin{bmatrix} -1 \\ 1 \\ -2 \end{bmatrix},$$

$W = \text{span}(\vec{u}_1, \vec{u}_2)$ and

$$\vec{y} = \begin{bmatrix} -9 \\ 1 \\ 6 \end{bmatrix}.$$

a. Show that $\vec{u}_1$ and $\vec{u}_2$ are orthogonal.

b. What are the closest points to $\vec{y}$ in W?

c. Find the distance between W and $\vec{y}$.

d. Convert $\{\vec{u}_1, \vec{u}_2\}$ into an orthonormal basis for W.

e. Compute the projection of $\vec{y}$ onto W using the $(*)$ formula given in Theorem 4.

f. Write $\vec{y}$ as a sum of two vectors one in W and the other in $W^{\perp}$.

9.6.3 The Gram–Schmidt Process

In the previous sections, we saw the importance of working with orthogonal and orthonormal bases. In this section, we explore *the Gram–Schmidt process* which is a simple algorithm to construct an orthogonal or orthonormal basis from any nonzero subspace of $\mathbb{R}^n$.

Idea: construction of an orthogonal basis from a basis with two elements

To see the idea behind the Gram–Schmidt process let's review Example 2 in Section 5.2:

Suppose $\vec{y} = \begin{bmatrix} 7 \\ 6 \end{bmatrix}$ and $\vec{u} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$. $\vec{y}$ and $\vec{u}$ are linearly independent and form a basis for $\mathbb{R}^2$. Clearly, as shown in below figure, $\vec{y}$ and $\vec{u}$ are not orthogonal, however $\vec{y}$ and $\vec{z}$ (the component of $\vec{y}$ orthogonal to $\vec{u}$) form an orthogonal basis for $\mathbb{R}^2$

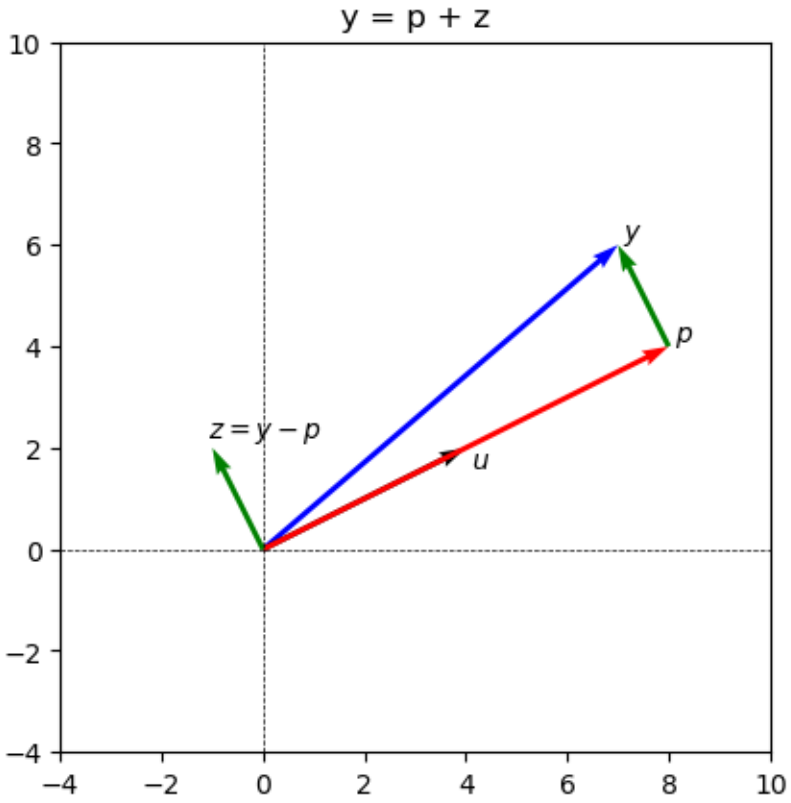
```
# setup vectors

y = np.array([7,6])

u = np.array([4,2])

# projection of y onto u

proj_y_u = (np.dot(y,u)/ np.dot(u,u))* u
print( "The projection of y onto u is ", proj_y_u)

z = y - proj_y_u

print("The component of y orthogonal to u is", z )


# Plot the coordinates in two separate plots

fig, ax = plt.subplots()

figsize=(10, 5)

# vector y
ax.quiver(0, 0, 7, 6, angles='xy', scale_units='xy', scale=1, color='blue', label ='y
↪')
ax.text(7.1,6.1,'$y$')

# vector u
ax.quiver(0, 0, 4, 2, angles='xy', scale_units='xy', scale=1, color='black')
ax.text(4.1,1.6,'$u$')

# orthogonal projection p
ax.quiver(0, 0, 8, 4, angles='xy', scale_units='xy', scale=1, color='red')
ax.text(8.1,4.1,'$p$')

# vector z
ax.quiver(0, 0, -1, 2, angles='xy', scale_units='xy', scale=1, color='green')
ax.text(-1.1,2.2,'$z = y - p$')

# A copy of vector z starting at the end of vector p
ax.quiver(8, 4, -1, 2, angles='xy', scale_units='xy', scale=1, color='green')
```

(continues on next page)

(continued from previous page)

```
ax.set_xlim([-4, 10])
ax.set_ylim([-4, 10])
ax.set_aspect('equal')
ax.axhline(y=0, color='black', linestyle='--', linewidth=0.5)
ax.axvline(x=0, color='black', linestyle='--', linewidth=0.5)

ax.set_title('y = p + z')


#plt.tight_layout()
plt.show()
```

```
The projection of y onto u is  [8. 4.]
The component of y orthogonal to u is [-1.  2.]
```



Therefore, we have converted a basis for $\mathbb{R}^2$ into an orthogonal basis for $\mathbb{R}^2$.

Construction of an orthogonal basis from a basis with three elements

Example 1

Let

$$\vec{x}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$

$$\vec{x}_2 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$

$$\vec{x}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix},$$

and

$$W = \text{span}(\{\vec{x}_1, \vec{x}_2, \vec{x}_3\}).$$

Construct an orthogonal basis for W.

Solution:

The set $\{\vec{x}_1, \vec{x}_2, \vec{x}_3\}$ is a linearly independent and forms a basis for W. We will construct an orthogonal set of vectors $\{\vec{v}_1, \vec{v}_2, \vec{v}_3\}$ such that $W = \text{span}(\{\vec{v}_1, \vec{v}_2, \vec{v}_3\})$.

Step 1:

Define $\vec{v}_1 := \vec{x}_1$ and $W_1 := \text{span}(\vec{v}_1)$.

Step 2: We seek a vector $\vec{v}_2$ orthogonal to $\vec{v}_1$. Project $\vec{x}_2$ onto W_1; the component of $\vec{x}_2$ orthogonal to the subspace W_1 has the desired property:

$$\vec{v}_2 = \vec{x}_2 - \text{proj}_{W_1}^{\vec{x}_2} \quad \implies \vec{v}_2 \quad \perp \vec{v}_1$$

Now, let $W_2 = \text{span}(\vec{v}_1, \vec{v}_2)$.

```
# setup vectors

x1 = np.array([1,1,1,1])

x2 = np.array([0,1,1,1])

x3 = np.array([0,0,1,1])

v1 = x1
# projection of x2 onto x1

proj_x2_v1 = (np.dot(x2,v1)/ np.dot(v1,v1))* v1
print( "The projection of x2 onto x1 is ", proj_x2_v1, '\n')
```

(continues on next page)

(continued from previous page)

```
v2 = x2 - proj_x2_v1

print("The component of x1 orthogonal to W1 is", v2 )
```

```
The projection of x2 onto x1 is  [0.75 0.75 0.75 0.75]

The component of x1 orthogonal to W1 is [-0.75  0.25  0.25  0.25]
```

Step 3:

We can use a similar approach to construct $\vec{v}_3$. We find the orthogonal projection of $\vec{x}_3$ onto W_2 and define $\vec{v}_3$ as the component of $\vec{x}_3$ that is orthogonal to the subspace W_2:

$$\vec{v}_3 = \vec{x}_3 - \text{proj}_{W_2}(\vec{x}_3)$$

Since both $\vec{x}_3$ and $\text{proj}_{W_2}(\vec{x}_3)$ are elements of W, $\vec{v}_3$ is also in W. Moreover, by definition, $\vec{v}_3$ is orthogonal to both $\vec{v}_1$ and $\vec{v}_2$. Thus, $\{\vec{v}_1, \vec{v}_2, \vec{v}_3\}$ forms an orthogonal basis for W.

```
# projection of x3 onto W2

proj_x3_W2 = (np.dot(x3,v1)/ np.dot(v1,v1))* v1  +  (np.dot(x3,v2)/ np.dot(v2,v2))* v2

print( "the projection of x3 onto W2 is ", proj_x3_W2, '\n')

# component of y orthogonal to p
v3 = x3 - proj_x3_W2

print("the component of x3 orthogonal to W2 is", v3 )
```

```
the projection of x3 onto W2 is  [0.         0.66666667 0.66666667 0.66666667]

the component of x3 orthogonal to W2 is [ 0.         -0.66666667  0.33333333  0.
 ↪33333333]
```

Step 4 (Optional)

Normalizing the basis elements $\vec{v}_1$, $\vec{v}_2$, and $\vec{v}_3$, we will get unit vectors $\vec{e}_1$, $\vec{e}_2$, and $\vec{e}_3$ respectively. $\{\vec{e}_1, \vec{e}_2, \vec{e}_3\}$ is an orthonormal basis for W.

```
# The unit vector of v1
e1 = v1 / np.sqrt(np.dot(v1,v1))

# The unit vector of v1
e2 = v2 / np.sqrt(np.dot(v2,v2))

# The unit vector of v1
e3 = v3 / np.sqrt(np.dot(v3,v3))

print("e1 = ", e1)
print("e2 = ", e2)
print("e3 = ", e3)
```

```
e1 =  [0.5 0.5 0.5 0.5]
e2 =  [-0.8660254   0.28867513  0.28867513  0.28867513]
e3 =  [ 0.         -0.81649658  0.40824829  0.40824829]
```

The Gram_Schmidt Process

Theorem 8 (the Gram-Schmidt algorithm)

Let $W \subset \mathbb{R}^n$ be a non-zero subspace, and $\{\vec{x_1}, \vec{x_2}, \dots, \vec{x_p}\}$ be a basis for W. Define:

$$\vec{v_1} = \vec{x_1} \quad \text{and} \quad W_1 = \text{span}(\vec{v_1}), \tag{9.11}$$
$$\vec{v_2} = \vec{x_2} - \text{proj}_{W_1}(\vec{x_2}) \quad \text{and} \quad W_2 = \text{span}(\{\vec{v_1}, \vec{v_2}\}), \tag{9.12}$$
$$\vdots \tag{9.13}$$
$$\vec{v_p} = \vec{x_p} - \text{proj}_{W_{p-1}}(\vec{x_p}) \quad \text{and} \quad W_p = \text{span}(\{\vec{v_1}, \vec{v_2}, \dots, \vec{v_3}\}). \tag{9.14}$$

Then $W = W_p$, and $\{\vec{v_1}, \vec{v_2}, \dots, \vec{v_p}\}$ is an orthogonal basis for W. Moreover, if $\vec{e_1}, \vec{e_2}, \dots, \vec{e_p}$ are the unit vectors of $\vec{v_1}, \vec{v_2}, \dots, \vec{v_p}$ respectively, then $\{\vec{e_1}, \vec{e_2}, \dots, \vec{e_p}\}$ is an orthonormal basis for W.

```
# setup A
A = np.transpose([x1, x2, x3])

# setup Q
Q = np.transpose([e1, e2, e3])

# Compute R= Q^T*A
R = np.transpose(Q) @ A
```

```
R
```

```
array([[2.00000000e+00, 1.50000000e+00, 1.00000000e+00],
       [1.11022302e-16, 8.66025404e-01, 5.77350269e-01],
       [1.11022302e-16, 1.11022302e-16, 8.16496581e-01]])
```

Note that elements below the diagonal in R are very small and close to zero. In fact, R should be an upper triangular matrix. The reason why Python won't display the exact decimal numbers we expect is that some decimal fractions cannot be represented exactly as binary fractions. To address this issue, we can set very small elements of R to zero.

```
eps = 0.000001
R[np.abs(R) < eps] = 0
R
```

```
array([[2.        , 1.5       , 1.        ],
       [0.        , 0.8660254 , 0.57735027],
       [0.        , 0.        , 0.81649658]])
```

Numerical Note

When the Gram–Schmidt process is run on a computer, round-off error can build up as the vectors $\vec{v_k}$ are calculated, one by one. Specifically, for large and unequal values of k and j, the dot products $\vec{v_k}^T \cdot \vec{u_j}$ may not be sufficiently close to zero, leading to a loss of orthogonality. This loss of orthogonality can be reduced substantially by rearranging the order of the calculations.

We can also utilize `numpy.linalg.qr` in Python to compute the QR factorization of a matrix. This function provides an efficient and accurate implementation of the Gram-Schmidt process avoiding unnecessary loss of orthogonality.

```
Q, R = np.linalg.qr(A)

print('Q = \n', Q, '\n\n')

print('R = \n', R)
```

```
Q =
 [[-0.5         0.8660254   0.        ]
 [-0.5        -0.28867513  0.81649658]
 [-0.5        -0.28867513 -0.40824829]
 [-0.5        -0.28867513 -0.40824829]]


R =
 [[-2.         -1.5        -1.        ]
 [ 0.         -0.8660254  -0.57735027]
 [ 0.          0.         -0.81649658]]
```

Exercises

Exercises

1. Let

$$\vec{x}_1 = \begin{bmatrix} 3 \\ 1 \\ -1 \\ 3 \end{bmatrix},$$

$$\vec{x}_2 = \begin{bmatrix} -5 \\ 1 \\ 5 \\ -7 \end{bmatrix},$$

$$\vec{x}_3 = \begin{bmatrix} 1 \\ 1 \\ -2 \\ 8 \end{bmatrix},$$

and

$$W = \text{span}(\{\vec{x}_1, \vec{x}_2, \vec{x}_3\}).$$

Construct an orthonormal basis for W.

2. Let A be a matrix whose columns are $\vec{x}_1$, $\vec{x}_2$, $\vec{x}_3$ from Exercise 1:

$$A = \begin{bmatrix} 3 & -5 & 1 \\ 1 & 1 & 1 \\ -1 & 5 & -2 \\ 3 & -7 & 8 \end{bmatrix}.$$

Find a QR decomposition for A.

3. Let $A = QR$, where Q is an $m \times n$ matrix with orthogonal columns, and R is an $n \times n$ matrix. Show that if the columns of A are linearly dependent, then R cannot be invertible.

9.6.4 Least-Squares Problems

Linear systems arising in applications are often inconsistent. In such situations, the best one can do is to find a vector $\vec{x}'$ that makes $A\vec{x}$ as close as possible to $\vec{b}$. We think of $A\vec{x}$ as an approximation of $\vec{b}$. The smaller $\|\vec{b} - A\vec{x}\|$, the better the approximation. Therefore, we are looking for a vector $\hat{y}$ such that $\|\vec{b} - A\hat{x}\|$ is as small as possible. Such $\vec{y}$ is called the *least square solution* of $A\vec{x} = \vec{b}$. The name is motivated by the fact that $\|\vec{b} - A\hat{x}\|$ is the square root of a sum of squares. In this section, we explore this idea further.

Let $A\vec{x} = \vec{b}$ be inconsistent, which implies $\vec{b} \notin \text{col}(A)$. Note that no matter what $\vec{x}$ is, $A\vec{x}$ lies in $\text{col}(A)$. From Section 5.2, we know that the closest point to $\vec{b}$ in $\text{col}(A)$ is the projection of $\vec{b}$ onto $\text{col}(A)$ (the best approximation problem). Let $\hat{b} = \text{proj}_{\text{col}(A)}(\vec{b})$. Since $A\vec{x} = \hat{b}$ is consistent, there are $\hat{x}$ such that $A\hat{x} = \hat{b}$. $\hat{x}$ is a least square solution of $A\vec{x} = \vec{b}$. Recall that $\vec{b} - \hat{b}$ is orthogonal to $\text{col}(A)$, and thus so is $\vec{b} - A\hat{x}$. In other words, $\vec{b} - A\hat{x}$ is orthogonal to each column of A, and we have:

$$A^T(\vec{b} - A\hat{x}) = 0 \quad \text{or} \quad A^T A\vec{x} = A^T\vec{b}.$$

The equation $A^T A\vec{x} = A^T\vec{b}$ is called the **normal equation** for $A\vec{x} = \vec{b}$.

Theorem 9

The set of least-squares solutions of $A\vec{x} = \vec{b}$ coincides with the nonempty set of solutions of the normal equation $A^T A\vec{x} = A^T\vec{b}$.

Example 1

Find a least-squares solution of the inconsistent system $A\vec{x} = \vec{b}$ for

$$A = \begin{bmatrix} 4 & 2 \\ 0 & 2 \\ 1 & 1 \end{bmatrix}$$

and

$$\vec{b} = \begin{bmatrix} 2 \\ 0 \\ 11 \end{bmatrix}.$$

Solution: First, let's find $A^T A$ and $A^T\vec{b}$:

```
A = np.array([[4,0], [0,2], [1,1]])

b = np.array([[2], [0], [11]])

# compute A^TA
ATA = A.transpose() @ A

print("A^TA = \n", ATA)

# compute A^Tb
ATb = A.transpose() @ b

print("\n A^Tb = \n", ATb)
```

```
A^TA =
 [[17  1]
 [ 1  5]]

 A^Tb =
 [[19]
 [11]]
```

Thus, the normal equation for $A\vec{x} = \vec{b}$ is given by:

$$\begin{bmatrix} 17 & 1 \\ 1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 19 \\ 11 \end{bmatrix}.$$

To solve this equation, we can use row operations; alternatively, since A^TA is invertible and 2×2, we can use the invertible matrix theorem. In many calculations, A^TA is invertible, but this is not always the case. In Theorem 2, we will see when this is true. The least square solution $\hat{x}$ is given by:

$$\hat{x} = (A^TA)^{-1}A^T\vec{b}.$$

```
# computing x_hat

x_hat = np.linalg.inv(ATA) @ ATb
x_hat
```

```
array([[1.],
       [2.]])
```

Example 2

Find a least-squares solution of the inconsistent system

$$A\vec{x} = \vec{b}$$

for

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

and

$$\vec{b} = \begin{bmatrix} -3 \\ -1 \\ 0 \\ 2 \\ 5 \\ 1 \end{bmatrix}.$$

Solution

Let's set up the normal equation for $A\vec{x} = \vec{b}$, and find a solution for it.

```
A = np.array([[1,1,0,0], [1,1,0,0], [1,0,1,0],[1,0,1,0], [1,0,0,1],[1,0,0,1]])

b = np.array([[-3,-1,0,2,5,1]])


# compute A^TA
ATA = A.T @ A
print('A^TA = \n', ATA)

# compute A^Tb
ATb = A.T @ b.T
print('\n A^Tb = \n', ATb)
```

```
A^TA =
 [[6 2 2 2]
 [2 2 0 0]
 [2 0 2 0]
 [2 0 0 2]]

 A^Tb =
 [[ 4]
 [-4]
 [ 2]
 [ 6]]
```

Thus, the normal equation is

$$\begin{bmatrix} 6 & 2 & 2 & 2 \\ 2 & 2 & 0 & 0 \\ 2 & 0 & 2 & 0 \\ 2 & 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 4 \\ -4 \\ 2 \\ 6 \end{bmatrix}$$

To solve this equation we form its augmented matrix:

```
#aumented
M = np.concatenate((ATA, ATb), axis=1)
M
```

```
array([[ 6,  2,  2,  2,  4],
       [ 2,  2,  0,  0, -4],
       [ 2,  0,  2,  0,  2],
       [ 2,  0,  0,  2,  6]])
```

Call row operations:

```
# Swap two rows

def swap(matrix, row1, row2):

    copy_matrix=np.copy(matrix).astype('float64')

    copy_matrix[row1,:] = matrix[row2,:]
    copy_matrix[row2,:] = matrix[row1,:]

    return copy_matrix
```

(continues on next page)

(continued from previous page)

```
# Multiply all entries in a row by a nonzero number


def scale(matrix, row, scalar):
    copy_matrix=np.copy(matrix).astype('float64')
    copy_matrix[row,:] = scalar*matrix[row,:]
    return copy_matrix

# Replacing a row1 by the sum of itself and a multiple of rpw2

def replace(matrix, row1, row2, scalar):
    copy_matrix=np.copy(matrix).astype('float64')
    copy_matrix[row1] = matrix[row1]+ scalar * matrix[row2]
    return copy_matrix
```

```
M1 = swap(M, 0, 2)
M1
```

```
array([[ 2.,  0.,  2.,  0.,  2.],
       [ 2.,  2.,  0.,  0., -4.],
       [ 6.,  2.,  2.,  2.,  4.],
       [ 2.,  0.,  0.,  2.,  6.]])
```

```
M2 = scale(M1, 0, 1/2)
M2
```

```
array([[ 1.,  0.,  1.,  0.,  1.],
       [ 2.,  2.,  0.,  0., -4.],
       [ 6.,  2.,  2.,  2.,  4.],
       [ 2.,  0.,  0.,  2.,  6.]])
```

```
M3 = replace(M2, 1, 0, -2)
M3
```

```
array([[ 1.,  0.,  1.,  0.,  1.],
       [ 0.,  2., -2.,  0., -6.],
       [ 6.,  2.,  2.,  2.,  4.],
       [ 2.,  0.,  0.,  2.,  6.]])
```

```
M4 = replace(M3, 2, 0, -6)
M4
```

```
array([[ 1.,  0.,  1.,  0.,  1.],
       [ 0.,  2., -2.,  0., -6.],
       [ 0.,  2., -4.,  2., -2.],
       [ 2.,  0.,  0.,  2.,  6.]])
```

```
M5 = replace(M4, 3, 0, -2)
M5
```

```
array([[ 1.,  0.,  1.,  0.,  1.],
       [ 0.,  2., -2.,  0., -6.],
```

(continues on next page)

(continued from previous page)

```
       [ 0.,  2., -4.,  2., -2.],
       [ 0.,  0., -2.,  2.,  4.]])
```

```
M6 = scale(M5, 1, 1/2)
M6
```

```
array([[ 1.,  0.,  1.,  0.,  1.],
       [ 0.,  1., -1.,  0., -3.],
       [ 0.,  2., -4.,  2., -2.],
       [ 0.,  0., -2.,  2.,  4.]])
```

```
M7 = replace(M6, 2, 1, -2)
M7
```

```
array([[ 1.,  0.,  1.,  0.,  1.],
       [ 0.,  1., -1.,  0., -3.],
       [ 0.,  0., -2.,  2.,  4.],
       [ 0.,  0., -2.,  2.,  4.]])
```

```
M8 = scale(M7, 2, -1/2)
M8
```

```
array([[ 1.,  0.,  1.,  0.,  1.],
       [ 0.,  1., -1.,  0., -3.],
       [-0., -0.,  1., -1., -2.],
       [ 0.,  0., -2.,  2.,  4.]])
```

```
M9 = replace(M8, 3, 2, 2)
M9
```

```
array([[ 1.,  0.,  1.,  0.,  1.],
       [ 0.,  1., -1.,  0., -3.],
       [-0., -0.,  1., -1., -2.],
       [ 0.,  0.,  0.,  0.,  0.]])
```

```
M10 = replace(M9, 0, 2, -1)
M10
```

```
array([[ 1.,  0.,  0.,  1.,  3.],
       [ 0.,  1., -1.,  0., -3.],
       [-0., -0.,  1., -1., -2.],
       [ 0.,  0.,  0.,  0.,  0.]])
```

```
M11 = replace(M10, 1, 2, 1)
M11
```

```
array([[ 1.,  0.,  0.,  1.,  3.],
       [ 0.,  1.,  0., -1., -5.],
       [-0., -0.,  1., -1., -2.],
       [ 0.,  0.,  0.,  0.,  0.]])
```

The general solution is $x_1 = 3 - x_4$, $x_2 = -5 + x_4$, $x_3 = -2 + x_4$, and x_4 is a free parameter. So the general

least-squares solution of $A\vec{x} = \vec{b}$ has the form:

$$\hat{x} = \begin{bmatrix} 3 \\ -5 \\ -2 \\ 0 \end{bmatrix} + x_4 \begin{bmatrix} -1 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

Any linear system $A\vec{x} = \vec{b}$ admits at least one least-squares solution (the orthogonal projection $\hat{b}$). For example, the least-squares solution of $A\vec{x} = \vec{b}$ in Example 1 was unique, while the linear system in Example 2 has infinitely many least-squares solutions.

The next theorem gives useful criteria for determining when there is only one least-squares solution.

Theorem 10

Let A be an $m \times n$ matrix. The following statements are equivalent:

1. The equation $A\vec{x} = \vec{b}$ has a unique least-squares solution for each $\vec{b} \in \mathbb{R}^n$.
2. The columns of A are linearly independent.
3. The matrix $A^T A$ is invertible.

In any of these cases, the least-squares solution $\hat{x}$ is given by:

$$\hat{x} = (A^T A)^{-1} A^T \vec{b}.$$

Moreover, if $A = QR$ is a QR-factorization of A, then the least-squares solution $\hat{x}$ is given by:

$$\hat{x} = R^{-1} Q^T \vec{b}. \quad (*)$$

Example 3

Let $A = \begin{bmatrix} 1 & 3 & 5 \\ 1 & 1 & 0 \\ 1 & 1 & 2 \\ 1 & 3 & 3 \end{bmatrix}$ and $\vec{b} = \begin{bmatrix} 3 \\ 5 \\ 7 \\ -3 \end{bmatrix}$. Find a least-squares solution of $A\vec{x} = \vec{b}$.

Solution:

A QR-factorization of A can be obtained as in Section 5.3 using `numpy.linalg.qr()`:

```
A = np.array([[1,3,5], [1,1,0], [1,1,2],[1,3,3]])

# QR factorization

Q, R = np.linalg.qr(A)

print('Q = \n', Q, '\n')

print('R = \n', R)
```

```
Q =
 [[-0.5  0.5 -0.5]
 [-0.5 -0.5  0.5]
```

(continues on next page)

(continued from previous page)

```
 [-0.5 -0.5 -0.5]
 [-0.5  0.5  0.5]]

R =
 [[-2. -4. -5.]
 [ 0.  2.  3.]
 [ 0.  0. -2.]]
```

Now we compute $\hat{x} = R^{-1}Q^T\vec{b}$:

```
# setup
b = np.array([[3, 5, 7, -3]]).T

#computing x_hat

x_hat = np.linalg.inv(R) @ Q.T @ b
x_hat
```

```
array([[10.],
       [-6.],
       [ 2.]])
```

Numerical Note

Since R in $(*)$ is an upper triangular matrix, we can alternatively compute $\hat{x}$ by finding the exact solutions of:

$$R\hat{x} = Q^T\vec{b}. \quad (**)$$

For large matrices, solving $(**)$ by back-substitution or row operations is faster than computing R^{-1} and using $(*)$.

Exercises

Exercises

1. Let

$$A = \begin{bmatrix} 1 & -3 & -3 \\ 1 & 5 & 1 \\ 1 & 1 & 2 \\ 1 & 7 & 2 \end{bmatrix}$$

and

$$\vec{b} = \begin{bmatrix} 5 \\ -3 \\ -5 \end{bmatrix}.$$

a. Find a least-squares solution of $A\vec{x} = \vec{b}$.

b. Compute the associated least-squares error $\|\vec{b} - A\hat{x}\|$.

2. Describe all least-squares solutions of the system

$$x + y = 2$$
$$x + y = 4$$

9.7 Solutions to Exercises

SECTION 9.2.2 Linear Systems

1. Find a linear system containing two equations in two unknowns that have infinitely many solutions. Verify your answer by plotting the lines of these equations on the same plane.

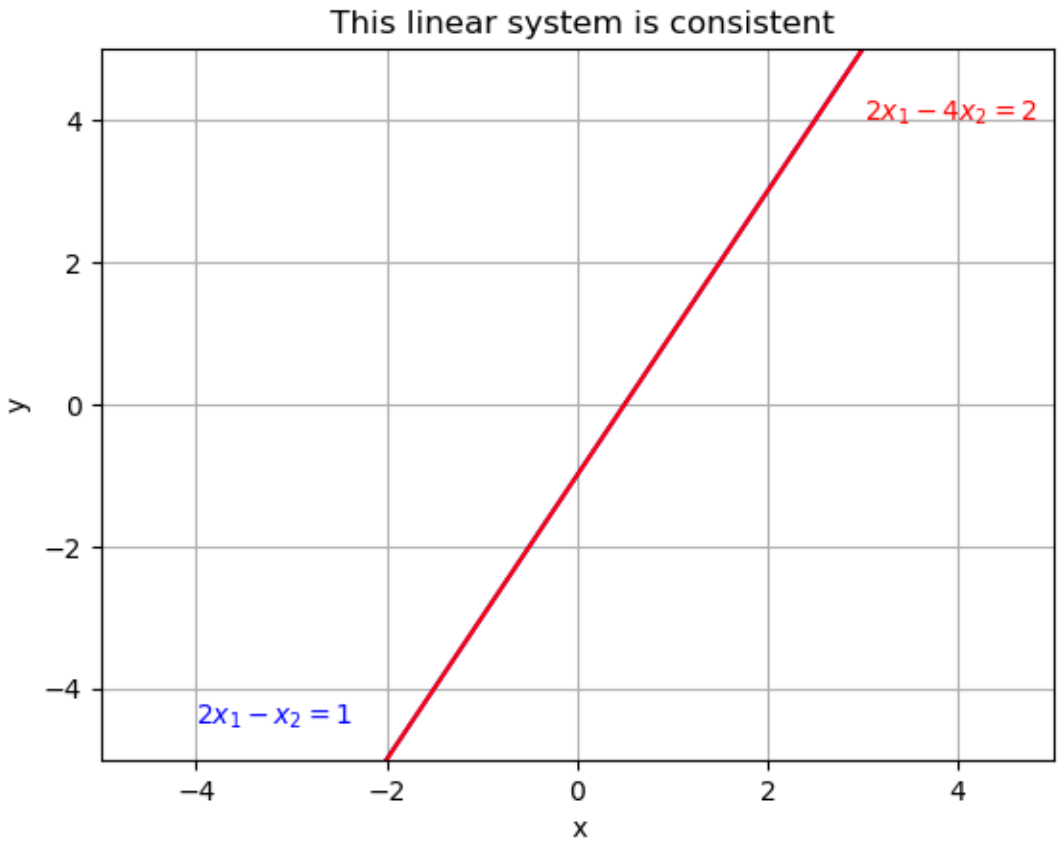
Solution:

The following system is consistent and has infinitely many solutions:

$$2x_1 - x_2 = 1$$
$$4x_1 - 2x_2 = 2$$

We expect that the graph of these two lines will coincide. Let's plot the graph of these lines on the same plane:

```
import matplotlib.pyplot as plt
import numpy as np

# Data for plotting

x = range(-5, 5)
y1 = [ 2*k-1 for k in x]
y2 = [ 2*k-1 for k in x]

fig, ax = plt.subplots()

# Specify the length of each axis
ax.set_xlim(-5,5)
ax.set_ylim(-5,5)

#Plot x and y1 using blue color
ax.plot(x, y1, color = 'b')
ax.text(-4,-4.5,'$2x_1 - x_2 = 1$', color = 'b')

#Plot x and y2 using red color
ax.plot(x, y2, color = 'r')
ax.text(3,4,'$2x_1 -4x_2 = 2$', color = 'r')


# Add labels and a title to the graph
ax.set(xlabel=' x', ylabel='y',
       title=' This linear system is consistent')
ax.grid()

plt.show()
```



2. Find a linear system that is equivalent to the linear system in Example 1.

Solution:

Any system that has the same solution set as the linear system in Example 1 is considered equivalent. Since the solution set is $\{(0, -1)\}$, the following system is an example of such a system:

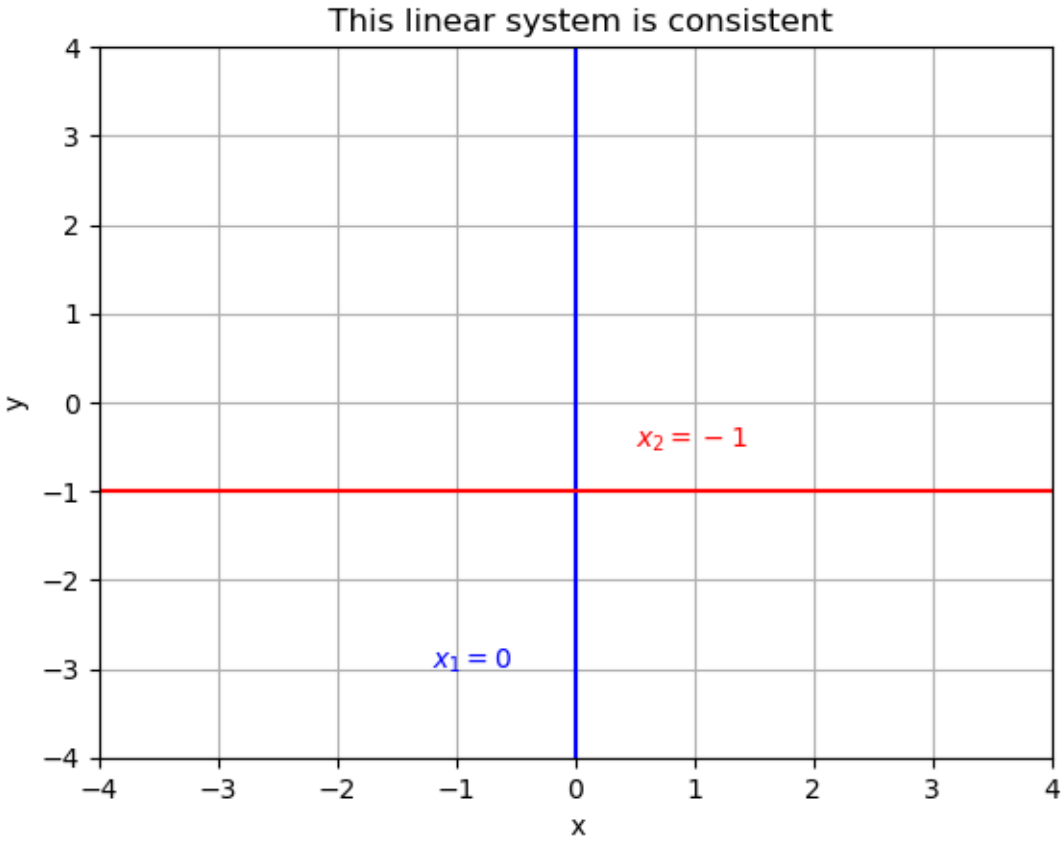
$$x_1 = 0$$
$$x_2 = -1$$

```
# Data for plotting

r = range(-5, 5)
x1 = [ 0 for k in r]
y1 = [-1 for k in r]

fig, ax = plt.subplots()

# Specify the length of each axis
ax.set_xlim(-4,4)
ax.set_ylim(-4,4)

#Plot x and y1 using blue color
ax.plot(x1, r, color = 'b')
ax.text(-1.2,-3,'$x_1 = 0$', color = 'b')

#Plot x and y2 using red color
```

(continues on next page)

(continued from previous page)

```
ax.plot(x, y1, color = 'r')
ax.text(0.5,-0.5, '$x_2 = -1$', color = 'r')


# Add labels and a title to the graph
ax.set(xlabel=' x', ylabel='y',
       title=' This linear system is consistent')
ax.grid()

plt.show()
```



3. Determine which of the following matrices are in REF. For those that are, identify their pivot columns. Are there any in RREF?

$$A = \begin{bmatrix} 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 3 \\ 4 & -3 & 8 \end{bmatrix}$$

A matrix is in row echelon form (REF) if it satisfies the following conditions:

1. The rows with all zeros must be at the bottom.
2. Every leading term of a row must be in a column to the right of the leading entry.
3. Entries below a leading entry must be zero.

Let's analyze each matrix and determine if they are in REF:

- Matrix A is not in REF because it does not satisfy the first condition (rows with all zeros must be at the bottom).
- Matrix B is in REF as it satisfies all the conditions.
- Matrix C is in REF as it satisfies all the conditions.

None of the matrices A, B, or C are in reduced row echelon form (RREF) because the conditions for RREF include having a pivot of 1 in every row, and both matrix A and matrix C have nonzero pivots other than 1.

4. For the above matrices, write the corresponding linear systems and find their solutions.

Solution:

A :

$$\begin{aligned} 4x_3 - 5x_4 &= 0 \\ 0 &= 0 \\ 0 &= 3 \end{aligned}$$

Clearly, because of the last equation, this system is inconsistent.

B :

$$\begin{aligned} x_1 + x_2 &= 1 \\ x_2 + x_3 &= 0 \\ x_3 &= 1 \end{aligned}$$

By back substitution, we can see that $x_1 = 2, -x_2 = x_3 = 1$.

C :

$$\begin{aligned} x_1 + 3x_3 &= 0 \\ x_2 &= 0 \\ 0 &= 1 \end{aligned}$$

Clearly, because of the last equation, this system is inconsistent.

5. Find the REF of $D = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 3 \\ 4 & -3 & 8 \end{bmatrix}$.

Solution

```
D = np.array([[0,1,2],[1,0,3],[4,-3,8]])

# Elementry row operations:

# Swap two rows

def swap(matrix, row1, row2):

    copy_matrix=np.copy(matrix).astype('float64')

    copy_matrix[row1,:] = matrix[row2,:]
    copy_matrix[row2,:] = matrix[row1,:]

    return copy_matrix


# Multiple all entries in a row by a nonzero number


def scale(matrix, row, scalar):
    copy_matrix=np.copy(matrix).astype('float64')
    copy_matrix[row,:] = scalar*matrix[row,:]
    return copy_matrix

# Replacing a row by the sum of itself and a multiple of another

def replace(matrix, row1, row2, scalar):
    copy_matrix=np.copy(matrix).astype('float64')
    copy_matrix[row1] = matrix[row1]+ scalar * matrix[row2]
    return copy_matrix
```

```
D1 = swap(D, 0, 1)
D1
```

```
array([[ 1.,  0.,  3.],
       [ 0.,  1.,  2.],
       [ 4., -3.,  8.]])
```

```
D2 = replace(D1, 2, 0, -4)
D2
```

```
array([[ 1.,  0.,  3.],
       [ 0.,  1.,  2.],
       [ 0., -3., -4.]])
```

```
D3 = replace(D2, 2, 1, 3)
D3
```

```
array([[1., 0., 3.],
       [0., 1., 2.],
       [0., 0., 2.]])
```

SECTION 9.2.3 Matrix Equations

1. Given the system of equations:

$$\begin{aligned} 3x + 2y &= 1 \\ x - 2y + z &= 5 \\ 2x - z &= 0 \end{aligned}$$

Write this system as:

1. a vector equation.
2. a matrix equation.

Solution:

-Vector Equation:

$$x \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix} + y \begin{bmatrix} 2 \\ -2 \\ 0 \end{bmatrix} + z \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 5 \\ 0 \end{bmatrix}$$

-Matrix Equation:

$$\begin{bmatrix} 3 & 2 & 0 \\ 1 & -2 & 1 \\ 2 & 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 \\ 5 \\ 0 \end{bmatrix}$$

2. True or False?

   (a). $\vec{0} \in$ span$\{\vec{v}_1, \vec{v}_2, \vec{v}_3\}$.

   (b). $\vec{v}_1 \in$ span$\{\vec{v}_1, \vec{v}_2, \vec{v}_3\}$.

   (c). span$\{\vec{v}_1, \vec{v}_2\} \subseteq$ span$\{\vec{v}_1, \vec{v}_2, \vec{v}_3\}$.

   (d). A set 4 vectors must be linearly DEPENDENT in $\mathbb{R}^3$.

Solution:

(a)

True. The zero vector is always in the span of any set of vectors. This is because the span of a set of vectors includes all possible linear combinations of those vectors, and

$$\vec{0} = 0\vec{v_1} + 0\vec{v_2} + 0\vec{v_3}$$

is a linear combination of $\vec{v_1}, \vec{v_2}$ and $\vec{v_3}$.

(b)

$\vec{v}_1 \in$ span$\{\vec{v}_1, \vec{v}_2, \vec{v}_3\}$:

True. Since $\vec{v}_1$ is a linear combination of $\vec{v}_1, \vec{v}_2$ and $\vec{v}_3\}$:

$$\vec{v_1} = 1\vec{v_1} + 0\vec{v_2} + 0\vec{v_3}$$

(c)

span$\{\vec{v}_1, \vec{v}_2\} \subseteq$ span$\{\vec{v}_1, \vec{v}_2, \vec{v}_3\}$:

True. Any linear combination of $\vec{v}_1$ and $\vec{v}_2$ is also a linear combination of $\vec{v}_1$, $\vec{v}_2$, and $\vec{v}_3$:

$$c_1\vec{v_1} + c_2\vec{v_2} = c_1\vec{v_1} + c_2\vec{v_2} + 0\vec{v_3} \in span\vec{v_1}, \vec{v_2}, \vec{v_3}$$

(d)

A set of 4 vectors must be linearly dependent in $\mathbb{R}^3$:

True. If we use the four vectors as columns of a matrix A, then A will be 3×4 which cannot have a pivot in every column. So the homogeneous system $A\vec{x} = 0$ has nontrivial solutions making the columns DEPENDENT.

SECTION 9.2.4 Solution Set of Linear Systems

1. Is

$$\left\{ \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}, \begin{bmatrix} 7 \\ 8 \\ 9 \end{bmatrix} \right\}$$

linearly independent or dependent?

Solution:

Let's denote the given vectors as:

$$\vec{v}_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \quad \vec{v}_2 = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}, \quad \vec{v}_3 = \begin{bmatrix} 7 \\ 8 \\ 9 \end{bmatrix}$$

Now, we'll set up the following linear combination and solve for the coefficients:

$$c_1\vec{v}_1 + c_2\vec{v}_2 + c_3\vec{v}_3 = \vec{0}$$

Substituting the given vectors and equating the resulting components to zero, we have the following linear system:

$$\begin{aligned} c_1 + 4c_2 + 7c_3 &= 0 \\ 2c_1 + 5c_2 + 8c_3 &= 0 \\ 3c_1 + 6c_2 + 9c_3 &= 0 \end{aligned}$$

$\{\vec{v_1}, \vec{v_2}\vec{v_3}\}$ is a linearly independent set is and only if this system has only the trivial solution. To verify this we set up the augmented matrix of this system and apply the row operations:

```
import numpy as np

A = np.array([[1, 4, 7, 0], [2, 5, 8, 0], [3, 6, 9, 0]])
A
```

```
array([[1, 4, 7, 0],
       [2, 5, 8, 0],
       [3, 6, 9, 0]])
```

```
A1 = replace(A, 1, 0, -2)
A1
```

```
array([[ 1.,  4.,  7.,  0.],
       [ 0., -3., -6.,  0.],
       [ 3.,  6.,  9.,  0.]])
```

```
A2 = replace(A1, 2, 0, -3)
A2
```

```
array([[  1.,   4.,   7.,   0.],
       [  0.,  -3.,  -6.,   0.],
       [  0.,  -6., -12.,   0.]])
```

```
A3 = replace(A2, 2, 1, -2)
A3
```

```
array([[ 1.,  4.,  7.,  0.],
       [ 0., -3., -6.,  0.],
       [ 0.,  0.,  0.,  0.]])
```

```
A4 = scale(A3, 1, -1/3)
A4
```

```
array([[ 1.,  4.,  7.,  0.],
       [-0.,  1.,  2., -0.],
       [ 0.,  0.,  0.,  0.]])
```

From see this we can see that c_3 is a free variable and the system has infinitely many solutions, and thus, $\{\vec{v_1}, \vec{v_2}\vec{v_3}\}$ is linearly dependent.

2. Describe the solution set of $A\vec{x} = \vec{b}$ where

$$A = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}, \quad \vec{b} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix},$$

Solution:

```
Ab = np.array([[1,4,7,1],[2,5,8,2],[3,6,9,3]])
Ab

# Swap two rows

def swap(matrix, row1, row2):

    copy_matrix=np.copy(matrix).astype('float64')

    copy_matrix[row1,:] = matrix[row2,:]
    copy_matrix[row2,:] = matrix[row1,:]

    return copy_matrix


# Multiple all entries in a row by a nonzero number


def scale(matrix, row, scalar):
    copy_matrix=np.copy(matrix).astype('float64')
    copy_matrix[row,:] = scalar*matrix[row,:]
    return copy_matrix

# Replacing a row1 by the sum of itself and a multiple of row2

def replace(matrix, row1, row2, scalar):
    copy_matrix=np.copy(matrix).astype('float64')
    copy_matrix[row1] = matrix[row1]+ scalar * matrix[row2]
    return copy_matrix
```

```
B1 = replace(Ab, 1 , 0, -2)
B1
```

```
array([[ 1.,  4.,  7.,  1.],
       [ 0., -3., -6.,  0.],
       [ 3.,  6.,  9.,  3.]])
```

```
B2 = replace(B1, 2 , 0, -3)
B2
```

```
array([[  1.,   4.,   7.,   1.],
       [  0.,  -3.,  -6.,   0.],
       [  0.,  -6., -12.,   0.]])
```

```
B3 = replace(B2, 2, 1 , -2)
B3
```

```
array([[ 1.,  4.,  7.,  1.],
       [ 0., -3., -6.,  0.],
       [ 0.,  0.,  0.,  0.]])
```

```
B4 = scale(B3, 1 , -1/3)
B4
```

```
array([[ 1.,  4.,  7.,  1.],
       [-0.,  1.,  2., -0.],
       [ 0.,  0.,  0.,  0.]])
```

From this we see that for any $z \in \mathbb{R}$ we have a unique solution. A general solution for this system is:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1+z \\ -2z \\ z \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} z \\ -2z \\ z \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + z \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

SECTION 9.3.1 Matrix Algebra

1. Which pair of the following matrices can be multiplied? Compute their matrix product.

$$A = \begin{bmatrix} 1 & 3 \\ 4 & -2 \\ 3 & 2 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 4 & 5 \\ 0 & 2 & 3 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 3 & 0 \\ 2 & -1 & 3 \\ 0 & 1 & 1 \end{bmatrix}$$

Solution:

```
import numpy as np

A = np.array([[1,3], [4,-2], [3,2]])

B = np.array([[1,4,5], [0,2,3]])

C = np.array([[1,3,0], [2,-1, 3], [0,1,1]])
```

None of AA, CB, BB, and AC can be performed because none of the neighboring dimensions match. For example, if we try to compute AA in Python we get the following error:

```
A@A
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[5], line 1
----> 1 A@A

ValueError: matmul: Input operand 1 has a mismatch in its core dimension 0, with␣
↪gufunc signature (n?,k),(k,m?)->(n?,m?) (size 3 is different from 2)
```

On the other hand AB, BC, CA, BA, CC can be performed:

```
A@B
```

```
array([[ 1, 10, 14],
       [ 4, 12, 14],
       [ 3, 16, 21]])
```

```
B@C
```

```
array([[ 9,  4, 17],
       [ 4,  1,  9]])
```

```
C@A
```

```
array([[13, -3],
       [ 7, 14],
       [ 7,  0]])
```

```
C@C
```

```
array([[ 7,  0,  9],
       [ 0, 10,  0],
       [ 2,  0,  4]])
```

2. Find two matrices A and B such that the products AB and BA are defined but $AB \neq BA$.

Solution:

```
A = np.array([[1, 2, 3], [4, 5, 6]])
B = np.array([[7, 8], [9, 10], [11, 12]])

AB = A@B
BA = B@A

print('AB = \n', AB)

print('\nBA = \n', BA)
```

```
AB =
 [[ 58  64]
 [139 154]]

BA =
```

(continues on next page)

(continued from previous page)

```
[[ 39  54  69]
[ 49  68  87]
[ 59  82 105]]
```

3. Given $A = \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix}$, find a nonzero matrix C for which $AC = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$.

Solution

There are infinitely many matrices such that $AC = 0$. We can set $C = \begin{bmatrix} c_1 & C_2 \\ C_3 & C_4 \end{bmatrix}$ and solve $AC = 0$ for C. For example one solution is $C = \begin{bmatrix} -2 & 1 \\ 1 & 0.5 \end{bmatrix}$

```
A = np.array([[1,2], [1,2]])

C = np.array([[-2,1], [1, -0.5]])


A@C
```

```
array([[0., 0.],
       [0., 0.]])
```

SECTION 9.3.2 Transpose, Inverse and Trace of matrices

1. Suppose $A = \begin{bmatrix} 1 & 2 \\ 0 & -1 \\ 3 & 0 \end{bmatrix}$

   (a) Compute A^T.

   (b) Is AA^T invertible?

   (c) Is $A^T A$ invertible?

Solution:

```
# part(a)

A = np.array([[1,2], [0,-1], [3,0]])

# compute transpose of A
A_T = np.transpose(A)

A_T
```

```
array([[ 1,  0,  3],
       [ 2, -1,  0]])
```

(b)

Solution 1 Obviously a 3×2 matrix has at most two linearly independent columns. It can be shown that AA^T has the same amount of linearly independent columns. Thus AA^T as a 3×3 matrix does not have three linearly independent columns and therefore, is not invertible by the invertible matrix theorem.

Solution 2 Lets check and see if AA^T row equivalent to identity matrix I_3

```
# part(b)

B = A @ A_T
B
```

```
array([[ 5, -2,  3],
       [-2,  1,  0],
       [ 3,  0,  9]])
```

We form the augmented matrix $[AA^T \mid I_3]$ and use row operations to check if it is invertible:

```
# 3x3 Identity matrix
I3 = np.eye(3)

# Augmented matrix [AA^T, I_3]
M = np.concatenate((B, I3), axis=1)
M
```

```
array([[ 5., -2.,  3.,  1.,  0.,  0.],
       [-2.,  1.,  0.,  0.,  1.,  0.],
       [ 3.,  0.,  9.,  0.,  0.,  1.]])
```

We can now use the following row operations to check if B is row equivalent to the identity matrix.

```
# Swap two rows

def swap(matrix, row1, row2):

    copy_matrix=np.copy(matrix).astype('float64')

    copy_matrix[row1,:] = matrix[row2,:]
    copy_matrix[row2,:] = matrix[row1,:]

    return copy_matrix


# Multiple all entries in a row by a nonzero number


def scale(matrix, row, scalar):
    copy_matrix=np.copy(matrix).astype('float64')
    copy_matrix[row,:] = scalar*matrix[row,:]
    return copy_matrix

# Replacing row1 by the sum of itself and a multiple of row2

def replace(matrix, row1, row2, scalar):
    copy_matrix=np.copy(matrix).astype('float64')
    copy_matrix[row1] = matrix[row1]+ scalar * matrix[row2]
    return copy_matrix
```

```
M1 = swap(M, 0, 1)
M1
```

```
array([[-2.,  1.,  0.,  0.,  1.,  0.],
       [ 5., -2.,  3.,  1.,  0.,  0.],
```

(continues on next page)

(continued from previous page)

```
       [ 3.,  0.,  9.,  0.,  0.,  1.]])
```

```
M2 = scale(M1, 0, -1/2)
M2
```

```
array([[ 1. , -0.5, -0. , -0. , -0.5, -0. ],
       [ 5. , -2. ,  3. ,  1. ,  0. ,  0. ],
       [ 3. ,  0. ,  9. ,  0. ,  0. ,  1. ]])
```

```
M3 = replace(M2, 1, 0, -5)
M3
```

```
array([[ 1. , -0.5, -0. , -0. , -0.5, -0. ],
       [ 0. ,  0.5,  3. ,  1. ,  2.5,  0. ],
       [ 3. ,  0. ,  9. ,  0. ,  0. ,  1. ]])
```

```
M4 = replace(M3, 2, 0, -3)
M4
```

```
array([[ 1. , -0.5, -0. , -0. , -0.5, -0. ],
       [ 0. ,  0.5,  3. ,  1. ,  2.5,  0. ],
       [ 0. ,  1.5,  9. ,  0. ,  1.5,  1. ]])
```

```
M5 = scale(M4, 1, 2)
M5
```

```
array([[ 1. , -0.5, -0. , -0. , -0.5, -0. ],
       [ 0. ,  1. ,  6. ,  2. ,  5. ,  0. ],
       [ 0. ,  1.5,  9. ,  0. ,  1.5,  1. ]])
```

```
M6 = replace(M5, 2, 1, -1.5)
M6
```

```
array([[ 1. , -0.5, -0. , -0. , -0.5, -0. ],
       [ 0. ,  1. ,  6. ,  2. ,  5. ,  0. ],
       [ 0. ,  0. ,  0. , -3. , -6. ,  1. ]])
```

From this we can see that it is impossible to get an identity matrix on the left side, so AA^T is not invertible.

(C):

Let's compute $A^T A$

```
C = A_T @ A
C
```

```
array([[10,  2],
       [ 2,  5]])
```

The determinant of C is $50 - 4 = 46$, not zero, and therefore, C is invertible.

2. The trace is invariant under cyclic permutations, i.e., for three matrices we have $Tr(ABC) = Tr(BCA) = Tr(CAB)$, where A is an $n \times p$ matrix, B is a $p \times m$ matrix, and C is an $m \times n$ matrix.

Using Example 9, generate three matrices A, B, and C such that the product ABC makes sense, and verify that:

$$Tr(ABC) = Tr(BCA) = Tr(CAB)$$

Solution

```
# generate a 5x3 matrix
A = np.random.rand(5, 3)

# generate a 3x2 matrix
B = np.random.rand(3, 2)


# generate a 2x5 matrix
C = np.random.rand(2, 5)

# find ABC
ABC = A @ B @ C

print('ABC = \n \n ', ABC)

# find BCA

BCA = B @ C @ A

print('\n BCA = \n \n ', BCA)

# find CAB

CAB = C @ A @ B

print('\n CAB = \n \n ', CAB)

#trace of ABC

Tr_ABC = np.trace(ABC)
print('\n Tr(ABC) = ', Tr_ABC)

Tr_BCA = np.trace(BCA)
print('\n Tr(BCA) = ', Tr_BCA)

Tr_CAB = np.trace(CAB)
print('\n Tr(CAB) = ', Tr_CAB)
```

```
ABC =

  [[0.98987463 1.14173923 0.9749678  0.54548799 1.17771615]
 [0.68823143 0.79423646 0.62002792 0.3285909  0.78062694]
 [0.89061397 1.02686076 0.93109763 0.53800509 1.09521993]
 [1.01585235 1.17163085 1.01045759 0.56847944 1.21516502]
 [1.32387689 1.52734839 1.25343552 0.68530018 1.54173913]]

 BCA =

  [[1.20700173 1.31688659 1.02761727]
 [2.75843882 3.01690466 2.4760114 ]
 [0.59342757 0.65299393 0.60152091]]
```

(continues on next page)

(continued from previous page)

```
CAB =

 [[2.1970437  2.34545965]
[2.3243565  2.6283836 ]]

Tr(ABC) =  4.825427299545087

Tr(BCA) =  4.825427299545087

Tr(CAB) =  4.825427299545087
```

SECTION 9.3.3 Determinant

1. Let A be an $n \times n$ matrix such that $A^2 = I_n$. Show that $det(A) = \pm 1$.

Solution:

By Theorem 4, part (2), $\det(AA) = \det(A) \cdot \det(A) = \det(A)^2$. Theorem 1, we have $\det(I_n) = 1$. Therefore,

$$\det(A)^2 = 1$$

Which means

$$\det(A) = \pm 1$$

2. Find the determinant of $A = \begin{bmatrix} 1 & 1 & 2 \\ 2 & 0 & 3 \\ 5 & -3 & 8 \end{bmatrix}$ using the row reduction method. Is A invertible?

Solution:

```
A = np.array([[1,1,2], [2,0,3], [5,-3,8]])

# Elementry row operations:

# Swap two rows

def swap(matrix, row1, row2):

    copy_matrix=np.copy(matrix).astype('float64')

    copy_matrix[row1,:] = matrix[row2,:]
    copy_matrix[row2,:] = matrix[row1,:]

    return copy_matrix


# Multiply all entries in a row by a nonzero number


def scale(matrix, row, scalar):
    copy_matrix=np.copy(matrix).astype('float64')
    copy_matrix[row,:] = scalar*matrix[row,:]
    return copy_matrix

# Replacing a row1 by the sum of itself and a multiple of row2

def replace(matrix, row1, row2, scalar):
```

(continues on next page)

(continued from previous page)

```
    copy_matrix=np.copy(matrix).astype('float64')
    copy_matrix[row1] = matrix[row1]+ scalar * matrix[row2]
    return copy_matrix
```

```
A
```

```
array([[ 1,  1,  2],
       [ 2,  0,  3],
       [ 5, -3,  8]])
```

```
A1 = replace(A, 1, 0, -2)
A1
```

```
array([[ 1.,  1.,  2.],
       [ 0., -2., -1.],
       [ 5., -3.,  8.]])
```

```
A2 = replace(A1, 2, 0, -5)
A2
```

```
array([[ 1.,  1.,  2.],
       [ 0., -2., -1.],
       [ 0., -8., -2.]])
```

```
A3 = replace(A2, 2, 1, -4)
A3
```

```
array([[ 1.,  1.,  2.],
       [ 0., -2., -1.],
       [ 0.,  0.,  2.]])
```

A_3 is already an upper-triangular matrix and we have

$$\det(A_3) = -4$$

Since we haven't used any swap operations we get:

$$\det(A) = \det(A_3) = -4$$

3. Use determinants to decide if the set $\left\{ \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}, \begin{bmatrix} 7 \\ 8 \\ 9 \end{bmatrix} \right\}$ is linearly independent.

Solution:

A matrix A is invertible if and only if $\det(A) \neq 0$ (Theorem 3). By the invertible matrix theorem, we know that A is invertible if and only if the columns of A are linearly independent.

```
# matrix A
A = np.array([[1,4, 7], [2,5,8], [3,6,9]])


# Compute a 2x2 determinant
def det2(matrix):
```

(continues on next page)

(continued from previous page)

```
    if matrix.shape == (2, 2):
        return matrix[0,0] * matrix[1,1] - matrix[0,1] * matrix[1,0]
    else:
        print("Please enter a 2x2 matrix")

# Computes the determinant of an n x n matrix.
def det(matrix):
    n = matrix.shape[0]
    if n == 1:
        return matrix[0, 0]
    elif n == 2:
        return matrix[0, 0] * matrix[1, 1] - matrix[0, 1] * matrix[1, 0]
    else:
        det = 0
        for j in range(n):
            submatrix = np.delete( (np.delete(matrix, 0 , 0)), j ,1)
            det += (-1) ** (0 + j) * matrix[0, j] * det2(submatrix)
        return det

print('det(A) = ', det(A))
```

```
det(A) =  0
```

4. Suppose A, B, P are $n \times n$ matrices, P is invertible, and r is a real number. Mark True or False:

   a. $det(A + B) = det(A) + det(B)$.

   b. $det(rA) = r \cdot det(A)$.

   c. $det(PAP^{-1}) = det(A)$.

Solution

a. False. For example, consider matrices:

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}.$$

Clearly, $\det(A) = 2$, $\det(B) = 3$, and $\det(A + B) = 10$.

b. False. For example, let $A = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$ and $r = 2$. Then, $\det(2A) = 8$, whereas $2\det(A) = 4$.

c. True. Using the invertibility of P and the properties of determinants, we have

$$\det(PAP^{-1}) = \det(P) \cdot \det(A \cdot P^{-1}) = \det(A \cdot P^{-1}) \cdot \det(P) = \det(A \cdot P^{-1} \cdot P) = \det(A)$$

5. Determine if the following points are coplanar (i.e., they lie on the same plane):

$$\begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 4 \\ 3 \\ 5 \end{bmatrix}, \begin{bmatrix} 3 \\ 0 \\ 6 \end{bmatrix}$$

Solution:

```
A = np.array([[1,4,3], [0,3,0], [2,5,6]])
A
```

```
array([[1, 4, 3],
       [0, 3, 0],
       [2, 5, 6]])
```

```
print('det(A) = ', det(A))
```

```
det(A) =  0
```

SECTION 9.4.1 Subspaces of $\mathbb{R}^n$

1. Determine which one of the following sets is a subspace:

   a. $H = \left\{ \begin{bmatrix} x \\ y \\ z \end{bmatrix} \in \mathbb{R}^3 \text{ with } x \geq 0, \right\}$.

   b. $H = \left\{ \begin{bmatrix} x \\ 1 \\ 2x \end{bmatrix} \in \mathbb{R}^3 \right\}$.

Solution

a. is not a subspace because it is not closed under addition. For example, let

$$\vec{u} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \text{and} \quad \vec{v} = \begin{bmatrix} -2 \\ 0 \\ 0 \end{bmatrix}$$

Then $\vec{u} + \vec{v}$ is not in H.

b. H is not a subspace because $\vec{o}$ is not in H.

2. Let $A = \begin{bmatrix} 1 & 2 & -9 \\ -2 & -2 & 2 \\ 2 & 3 & 0 \end{bmatrix}$. Find a basis for $col(A)$ and $null(A)$.

Solution:

We use row operation to find the pivot columns of A:

```
import numpy as np

A = np.array([[1,2,-9], [-2,-2,2],[2,3,0]])

# Elementry row operations:

# Swap two rows

def swap(matrix, row1, row2):

    copy_matrix=np.copy(matrix).astype('float64')

    copy_matrix[row1,:] = matrix[row2,:]
    copy_matrix[row2,:] = matrix[row1,:]

    return copy_matrix


# Multiply all entries in a row by a nonzero number
```

(continues on next page)

(continued from previous page)

```
def scale(matrix, row, scalar):
    copy_matrix=np.copy(matrix).astype('float64')
    copy_matrix[row,:] = scalar*matrix[row,:]
    return copy_matrix

# Replacing a row1 by the sum of itself and a multiple of row2

def replace(matrix, row1, row2, scalar):
    copy_matrix=np.copy(matrix).astype('float64')
    copy_matrix[row1] = matrix[row1]+ scalar * matrix[row2]
    return copy_matrix
```

```
A
```

```
array([[ 1,  2, -9],
       [-2, -2,  2],
       [ 2,  3,  0]])
```

```
A1 = replace(A, 1, 0, 2)
A1
```

```
array([[  1.,   2.,  -9.],
       [  0.,   2., -16.],
       [  2.,   3.,   0.]])
```

```
A2 = replace(A1, 2, 0, -2)
A2
```

```
array([[  1.,   2.,  -9.],
       [  0.,   2., -16.],
       [  0.,  -1.,  18.]])
```

```
A3 = scale(A2, 1, 0.5)
A3
```

```
array([[ 1.,  2., -9.],
       [ 0.,  1., -8.],
       [ 0., -1., 18.]])
```

```
A4 = replace(A3, 2, 1, 1)
A4
```

```
array([[ 1.,  2., -9.],
       [ 0.,  1., -8.],
       [ 0.,  0., 10.]])
```

From this we can see that all columns are pivots and the columns of A form a basis for col(A):

$$\left\{ \begin{bmatrix} 1 \\ 2 \\ -9 \end{bmatrix}, \begin{bmatrix} -2 \\ -2 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 \\ 3 \\ 0 \end{bmatrix} \right\}$$

Therefore, $dim(\text{col}(A) = 3)$. By Rank-Nulity Theorem, $dim(\text{null}(A) = 0)$, or in other words, $\text{null}(A) = \vec{o}$. We could also form the augmented matrix $[A|0]$ and find the RREF of it:

```
# zero vector
O = np.array([np.zeros(3)]).T
O
```

```
array([[0.],
       [0.],
       [0.]])
```

```
# Augmented matrix [A|0]

B = np.concatenate((A, O) , axis = 1)
B
```

```
array([[ 1.,  2., -9.,  0.],
       [-2., -2.,  2.,  0.],
       [ 2.,  3.,  0.,  0.]])
```

```
B1 = replace(B, 1, 0 , 2)
B1
```

```
array([[  1.,   2.,  -9.,   0.],
       [  0.,   2., -16.,   0.],
       [  2.,   3.,   0.,   0.]])
```

```
B2 = replace(B1, 2, 0, -2)
B2
```

```
array([[  1.,   2.,  -9.,   0.],
       [  0.,   2., -16.,   0.],
       [  0.,  -1.,  18.,   0.]])
```

```
B3 = scale(B2, 1, 0.5)
B3
```

```
array([[ 1.,  2., -9.,  0.],
       [ 0.,  1., -8.,  0.],
       [ 0., -1., 18.,  0.]])
```

```
B4 = replace(B3, 2, 1, 1)
B4
```

```
array([[ 1.,  2., -9.,  0.],
       [ 0.,  1., -8.,  0.],
       [ 0.,  0., 10.,  0.]])
```

```
B5 = scale(B4, 2, 1/10)
B5
```

```
array([[ 1.,  2., -9.,  0.],
       [ 0.,  1., -8.,  0.],
       [ 0.,  0.,  1.,  0.]])
```

```
B6 = replace(B5, 0, 1, -2)
B6
```

```
array([[ 1.,  0.,  7.,  0.],
       [ 0.,  1., -8.,  0.],
       [ 0.,  0.,  1.,  0.]])
```

```
B7 = replace(B6, 0, 2, -7)
B7
```

```
array([[ 1.,  0.,  0.,  0.],
       [ 0.,  1., -8.,  0.],
       [ 0.,  0.,  1.,  0.]])
```

```
B8 = replace(B7, 1, 2, 8)
B8
```

```
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.]])
```

3. Suppose A is a 5×5 matrix, and A has 3 pivot columns.
   a. Prove that $A\vec{x} = \vec{0}$ must have nontrivial solutions.
   b. Is there an example of A where $col(A) = null(A)$?

Solution:

a. By Rank-Nullity Theorem, $dim(\text{null}(A) = 2)$ so it is not trivial.

b. No, it is not because in such a matrix we would have

$$dim(col(A)) + dim(null(A)) = 5 \quad \text{which implies} \quad dim(col(A)) + dim(col(A)) = 5 \quad \text{or} \quad dim(col(A)) = 2.5$$

which is not possible.

4. Find an explicit example of a 4×4 matrix A which has $colA = null(A)$

Solution:

Let

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Clearly, $\vec{u} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ and $\vec{v} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$ form a basis for col(A). We can easily verify that $A\vec{u} = A\vec{v} = \vec{0}$, indicating that $\vec{u}$ and $\vec{v}$ are in $null(A)$. Now, we will show that they also form a basis for $null(A)$.

According to the rank-nullity theorem, we know that dim$(null(A)) = 2$. Thus, if we show that $\vec{u}$ and $\vec{v}$ are linearly independent, we are done! Fortunately, they are linearly independent because they are not multiples of each other.

SECTION 9.4.2 Introduction to Linear Transformations

1. Determine if each transformation is linear

a. $T : \mathbb{R}^2 \to \mathbb{R}^2$ given by

$$T\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_2 \\ -x_1 \end{bmatrix}.$$

b. $H : \mathbb{R}^2 \to \mathbb{R}^2$ given by

$$T\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} (x_1)^2 \\ 0 \end{bmatrix}.$$

Solution:

(a): T is a linear transformation since

$$T\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right) = T\left(\begin{bmatrix} x_1 + y_1 \\ x_2 + y_2 \end{bmatrix}\right) = \begin{bmatrix} x_2 + y_2 \\ -(x_1 + y_1) \end{bmatrix} = \begin{bmatrix} x_2 + y_2 \\ -x_1 - y_1 \end{bmatrix} = \begin{bmatrix} x_2 \\ -x_1 \end{bmatrix} + \begin{bmatrix} y_2 \\ -y_1 \end{bmatrix} = T\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) + T\left(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right)$$

and

$$T\left(c\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = T\left(\begin{bmatrix} cx_1 \\ cx_2 \end{bmatrix}\right) = \begin{bmatrix} -cx_2 \\ cx_1 \end{bmatrix} = c\begin{bmatrix} -x_2 \\ x_1 \end{bmatrix} = cT\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right)$$

(b) H is not a linear transformation because

$$cT\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = c\begin{bmatrix} (x_1)^2 \\ 0 \end{bmatrix} = \begin{bmatrix} c(x_1)^2 \\ 0 \end{bmatrix}$$

but

$$T\left(c\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = T\left(\begin{bmatrix} cx_1 \\ cx_2 \end{bmatrix}\right) = \begin{bmatrix} (cx_1)^2 \\ 0 \end{bmatrix} = \begin{bmatrix} (c^2(x_1)^2 \\ 0 \end{bmatrix}.$$

2. Find the standard matrix of $T : \mathbb{R}^2 \to \mathbb{R}^2$ given by $T(\vec{x}) =$ first reflect $\vec{x}$ across the x-axis and then rotate 45° counterclockwise.

Solution:

To find the standard matrix representation of T we apply T to the standard basis vectors of $\mathbb{R}^2$:

T reflects $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ to itself and then rotates it to $\begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{bmatrix}$.

T reflects $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ to $\begin{bmatrix} 0 \\ -1 \end{bmatrix}$ and then rotates it to $\begin{bmatrix} \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} \end{bmatrix}$.

Therefore, the standard matrix of T is $\begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix}$.

3. Suppose $T : \mathbb{R}^2 \to \mathbb{R}^3$ is a linear map and

$$T\left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad \text{and} \quad T\left(\begin{bmatrix} 0 \\ 1 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}.$$

Find $T\begin{bmatrix} 2 \\ 3 \end{bmatrix}$.

Solution

Note that

$$\begin{bmatrix} 2 \\ 3 \end{bmatrix} = 2\begin{bmatrix} 1 \\ 0 \end{bmatrix} + 3\begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Now

$$T\begin{bmatrix}2\\3\end{bmatrix} =$$

$$T\left(2\begin{bmatrix}1\\0\end{bmatrix} + 3\begin{bmatrix}0\\1\end{bmatrix}\right) =$$

$$2T\left(\begin{bmatrix}1\\0\end{bmatrix}\right) + 3T\left(\begin{bmatrix}1\\0\end{bmatrix}\right) =$$

$$2\begin{bmatrix}1\\2\\3\end{bmatrix} + 3\begin{bmatrix}0\\1\\-1\end{bmatrix} = \begin{bmatrix}2\\7\\3\end{bmatrix}$$

9.5.3 Eigenspace and characteristic polynomial

1. If $\vec{v}$ is an eigenvector of A corresponding to λ,what is $A^3\vec{v}$.

Solution:

$$A^3\vec{v} = A^2A\vec{v} = A^2\lambda\vec{v} = \lambda A^2\vec{v} = \lambda AA\vec{v} = \lambda A\lambda\vec{v} = \lambda^2 A\vec{v} = \lambda^3\vec{v}$$

2. If A is an $n \times n$ matrix and 2 is an eigenvalue of A, show that 4 is an eigenvalue of $2A$.

Solution:

Let $\vec{v}$ be an eigenvector of A corresponding to $\lambda = 2$. We have:

$$(2A)\vec{v} = 2A\vec{v} = 2(2\vec{v}) = 4\vec{v}$$

3. If A is an invertible matrix with eigenvalue λ show that A^{-1} has eigenvalue $1/\lambda$.

Solution:

Let $\vec{v}$ be an eigenvector of A corresponding to λ. We have

$$A\vec{v} = \lambda\vec{v} \iff A^{-1}(A\vec{v}) = A^{-1}(\lambda\vec{v}) \iff \vec{v} = \lambda A^{-1}\vec{v} \iff A^{-1}\vec{v} = \frac{1}{\lambda}\vec{v}.$$

4. Let $A = \begin{bmatrix}1 & 3 & 3\\-3 & -5 & -3\\3 & 3 & 1\end{bmatrix}$.

   a. Find the eigenvalues of A and their multiplicities.

   b. For each eigenvalue in part (a) find the corresponding eigenspace.

   c. Find the dimension of each eigenspace in part (b) by finding a basis for it. How this result is related to your answer to part (a)?

Solution:

```
import numpy as np

#(a)

A = np.array([[1,3,3], [-3, -5, -3], [3,3,1]])

#eigenvalues of A
np.linalg.eigvals(A)
```

```
array([ 1., -2., -2.])
```

(b)

$$E_1 = Null(\begin{bmatrix} 1-1 & 3 & 3 \\ -3 & -5-1 & -3 \\ 3 & 3 & 1-1 \end{bmatrix})$$

and

$$E_{-2} = Null(\begin{bmatrix} 1+2 & 3 & 3 \\ -3 & -5+2 & -3 \\ 3 & 3 & 1+2 \end{bmatrix})$$

(c) A basis for E_1 : $\left\{ \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} \right\}$. The dimension of E_1 is then 1.

A basis for E_{-2} : $\left\{ \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \right\}$

The dimension of E_{-2} is then 2.

The multiplicity of $\lambda = 1$ is equal to the dimension of E_1. The multiplicity of $\lambda = -2$ is equal to the dimension of E_{-2}.

In general the multiplicity of an eigenvalue λ is not equal to the dimension of its eigenspace E_λ. However, when it does we say the matrix is diagonalizable. We will learn more about this in the next section.

SECTION 9.5.4 Diagonalization

1. Suppose $M = \begin{bmatrix} 2 & 4 & 3 \\ -4 & -6 & -3 \\ 3 & 3 & 1 \end{bmatrix}$.

   a. Is M diagonalizable? If yes find a diagonalization for it.

   b. Use this diagonalization to compute $M^(100)$

Solution:

(a)

```
M = np.array([[2,4,3], [-4,-6,3], [3,3,1]])
# find evalues and evectors
evalues, evectors = np.linalg.eig(M)

print("eigenvalues = ",  evalues)
```

```
eigenvalues =  [-5. -2.  4.]
```

Since we have three distinct eigenvalues, M is diagonalizable.

```
lambda_1 = evalues[0]
lambda_2 = evalues[1]
lambda_3 = evalues[2]

# finding the diagonal matrix D
D = np.array([[lambda_1, 0, 0], [0, lambda_2, 0], [0, 0, lambda_3]])
```

(continues on next page)

(continued from previous page)

```
# finding the invirtible matrix P
P = evectors

# finding the inverse of matrix P

Q = np.linalg.inv(P)

#sanity check: A = PDQ

P @ D @ Q
```

```
array([[ 2.,  4.,  3.],
       [-4., -6.,  3.],
       [ 3.,  3.,  1.]])
```

(b): Note that

$$A^{100} = PD^{100}Q$$

```
# finding D^10

## digonal entries to the power of 10 (rounded it up to the closest intiger)
a1 = np.rint(pow(evalues[0], 100))
a2 = np.rint(pow(evalues[1], 100))
a3 = np.rint(pow(evalues[2], 100))

# computing D to the power of 10
D100 = np.array([[ a1, 0, 0], [0, a2, 0], [0, 0, a3]])

print(D100)

A100 = P @ D100 @ Q

A100
```

```
[[7.88860905e+69 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 1.26765060e+30 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 1.60693804e+60]]
```

```
array([[-4.38256058e+69, -4.38256058e+69,  4.38256059e+69],
       [ 9.64163329e+69,  9.64163329e+69, -9.64163329e+69],
       [-2.62953635e+69, -2.62953635e+69,  2.62953635e+69]])
```

2. Suppose $C = \begin{bmatrix} 3 & 0 & 1 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$. Is C diagonalizable?

Solution:

No. The eigenvalues of C are 3 with multiplicity 3 and 2 with multiplicity 2. We show C is not diagonalizable by looking for the dimension of $E_3 = Nul(C - 3I)$.

```
C = np.array([[3,0,1, 0, 0], [0,3,0,0,0], [0,0,3,0,0], [0,0,3,2,0], [0,0,0,0,2]])
```

(continues on next page)

(continued from previous page)

```
# find eigenvalues and eigenvectors
evalues, evectors = np.linalg.eig(C)

print('eigenvalues = ',evalues)

#identity matrix
I = np.eye(5)

# constructing C-3I
C - 3*I
```

```
eigenvalues =  [3. 3. 2. 3. 2.]
```

```
array([[ 0.,  0.,  1.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  3., -1.,  0.],
       [ 0.,  0.,  0.,  0., -1.]])
```

From this, we can see that the number of non-pivot columns in E_3 is 2 which is not equal to 3. Thus C is not diagonalizable.

3. Let A be a 5×5 matrix, with two eigenvalues. One eigenspace is three-dimensional, the other is two dimensional. Is A diagonalizable?

Solution:

Yes, since we have 5 linearly independent eigenvectors of A.

4. Compute A^{200} where $A = \begin{bmatrix} 4 & -3 \\ 2 & -1 \end{bmatrix}$.

Solution:

We can easily check that A has two distinct eigenvalues and therefore diagonalizable. Let's find a diagonalization for it.

```
A = np.array([[1,2], [4,3]])

# find evalues and evectors
evalues, evectors = np.linalg.eig(A)

lambda_1 = evalues[0]
lambda_2 = evalues[1]

# finding the diagonal matrix D
D = np.array([[lambda_1, 0], [0, lambda_2]])

# finding the invirtible matrix P
P = evectors

# finding the inverse of matrix P

Q = np.linalg.inv(P)

#sanity check: A = PDQ

P @ D @ Q
```

```
array([[1., 2.],
       [4., 3.]])
```

```
# finding D^200

a1 = np.rint(pow(evalues[0], 200))
a2 = np.rint(pow(evalues[1], 200))

# computing D to the power of 20
D200 = np.array([[ a1, 0], [0, a2]])

# computing A to the power of 20

A200 = P @ D200 @ Q

A200
```

```
array([[2.07433843e+139, 2.07433843e+139],
       [4.14867685e+139, 4.14867685e+139]])
```

SECTION 9.5.5 Discrete Dynamical Systems

1. Let A be a 2×2 matrix with eigenvalues 5 and $\frac{1}{5}$ and corresponding eigenvectors $\begin{bmatrix}1\\1\end{bmatrix}$ and $\begin{bmatrix}-1\\1\end{bmatrix}$. Let $\vec{x}_k$ be a solution of $\vec{x}_{k+1} = A\vec{x}_k, \vec{x}_0 = \begin{bmatrix}9\\1\end{bmatrix}$.

a. Compute $\vec{x}_1 = A\vec{x}_0$

b. Find a formula for $\vec{x}_k$ involving k and the eigenvectors of A.

c. Describe what happens when $k \to \infty$.

Solution:

```
#(a)

# initial vector
x_0 = np.array([9,1])

# matrix A
A = np.array([[1,-1],[1,1]])

#x_1
x_1 = A @ x_0
x_1
```

```
array([ 8, 10])
```

(b)

Assume

$$\vec{x_0} = d_1\vec{v_1} + d_2\vec{v_2} \quad \text{for some } d_1, d_2 \quad (*)$$

Then

$$\vec{x_k} = d_1(\lambda_1)^k\vec{v_1} + d_2(\lambda_2)^k\vec{v_2}$$

```
# a matrix whose columns are eigenvectors
P = np.array([[1,-1], [1,1]])

#solving (*) for coeeficients d1 and d2

D = np.linalg.solve(P, x_0)

print('d1 = ', D[0], ' and d2 = ',D[1] )
```

```
d1 =  5.0  and d2 =  -4.0
```

Therefore, we have

$$\vec{x_k} = 5(5)^k \begin{bmatrix} 1 \\ 1 \end{bmatrix} - 4(\frac{1}{5})^k \vec{v_2} \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

(C)

$$x_{k+1} \approx 5(5)^{k+1} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

SECTION 9.6.1 Inner product and Orthogonality

1. Let $\vec{a} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$.

   a. Find a vector $\vec{w}$ that is in the opposite direction of $\vec{a}$ and has a magnitude of 2.

   b. Find two non-parallel vectors $\vec{u}$ and $\vec{v}$ which are both orthogonal to $\vec{a}$

Solution

(a) we first find the unit vector of $\vec{a}$ and then multiply it by scaling factor -2.

```
import numpy as np

a = np.array([1,2,3,4])

#length of a
len_a = np.sqrt(np.dot(a,a))

#unit vector of a
e_a = a/len_a

#multyplying by scaling factor -2
w = -2 * e_a
```

(b) There are many options. Here are a few examples to consider:

$$\vec{u} = \begin{bmatrix} -2 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad \vec{v} = \begin{bmatrix} 0 \\ 0 \\ -4 \\ 3 \end{bmatrix}, \quad \vec{w} = \begin{bmatrix} -3 \\ 0 \\ 1 \\ 0 \end{bmatrix}.$$

2. Find two non-parallel vectors $\vec{u}$ and $\vec{v}$, which are both orthogonal to $\vec{a} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$ and $\vec{b} = \begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix}$.

Solution: It is not possible.

Suppose $\vec{u}$ and $\vec{v}$ exist, and $W = \text{span}(\vec{a}, \vec{b})$. Note that $dim(W) = 2$, and we have that $\vec{u}, \vec{v} \in W^{\perp}$. However, since $\vec{u}$ and $\vec{v}$ are not parallel, they are linearly independent. This means that the $dim(W^{\perp}) \geq 2$, which is a contradiction because, by Theorem 2, we should have $dim(W) + dim(W^{\perp}) = 3$.

3. Let $\vec{v} = \begin{bmatrix} 1 \\ 2 \\ 7 \end{bmatrix}$ and $\vec{w} = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$.

   a. Find $\vec{z} = \vec{v} - \left(\dfrac{\vec{v} \cdot \vec{w}}{\vec{w} \cdot \vec{w}} \right) \vec{w}$.

   b. Check that $\vec{z}$ is orthogonal to $\vec{w}$.

Solution:

```
# (a)

v = np.array([1,2,7])

w = np.array([1,-1,1])

##computing z
z = v - (np.dot(v,w)/np.dot(w,w)) * w

print('z = ', z, '\n')

#(b)

##checking z is orthogonal to w

print( 'z.w = ',np.dot(z,w))
```

```
z =  [-1.  4.  5.]

z.w =  0.0
```

4. If $W = span \left\{ \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \right\}$ find a basis for $W^{\perp}$.

Solution: We find a basis for W by looking for all solutions to the linear system:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \vec{0}.$$

The system of 1 linear equation in 4 unknowns has 3 free variables, x_2, x_3, x_4 with $x_1 = -2x_2 - 3x_3 - 4x_4$. So solutions look like:

$$\begin{bmatrix} -2x_2 - 3x_3 - 4x_4 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = x_2 \begin{bmatrix} -2 \\ 1 \\ 0 \\ 0 \end{bmatrix} + x_3 \begin{bmatrix} -3 \\ 0 \\ 1 \\ 0 \end{bmatrix} + x_4 \begin{bmatrix} -4 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

Thus a basis for $W^{\perp}$ is

$$\left\{ \begin{bmatrix} -2 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -3 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -4 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right\}.$$

8SECTION 9.6.2 Orthogonal Projection

1. Let $B = \left\{ \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right\}$ be a basis of $\mathbb{R}^2$.
   a. Check that this is an orthogonal basis.
   b. Find the coordinates of the vector $\begin{bmatrix} 3 \\ 5 \end{bmatrix}$ with respect to the basis B.

Solution:

```
#(a)

# Representing B as a matrix

B = np.array([[1,1], [1,-1]])

#Cheking the orthogonality of basis elements finding the dot product

np.dot(B[:,0], B[:,1])
```

```
0
```

```
# (b)

x = np.array([3,5])
#computing the coordinates of x with respect to B
x_B = np.linalg.inv(B) @ x

x_B
```

```
array([ 4., -1.])
```

2. Suppose $\vec{u}_1 = \begin{bmatrix} -7 \\ 1 \\ 4 \end{bmatrix}, \vec{u}_2 = \begin{bmatrix} -1 \\ 1 \\ -2 \end{bmatrix}, W = \text{span}(\vec{u}_1, \vec{u}_2)$ and $\vec{y} = \begin{bmatrix} 1 \\ 1 \\ 6 \end{bmatrix}$.
   a. Show that $\vec{u}_1$ and $\vec{u}_2$ are orthogonal.
   b. What is the closest points to $\vec{y}$ in W.
   c. Find the distance between W and $\vec{y}$.
   d. Convert $\{\vec{u}_1, \vec{u}_2\}$ into an orthonormal basis for W.
   e. Compute the projection of $\vec{y}$ onto W using the formula given in Theorem 4.
   f. Write $\vec{y}$ as sum of two vectors one in W and the other in $W^{\perp}$.

```
# (a)

u1 = np.array([-7,1,4])
```

(continues on next page)

(continued from previous page)

```
u2 = np.array([-1,1,-2])

# checking orthogonality using the dot product
np.dot(u1,u2)
```

```
0
```

```
# (b)

y = np.array([1,1,6])

# Finding the projection onto W (closest point to W)
Py_W = (np.dot(y,u1)/np.dot(u1,u1))* u1 + (np.dot(y,u2)/np.dot(u2,u2))* u2

Py_W
```

```
array([ 0.09090909, -1.72727273,  5.09090909])
```

```
# (c)

#computing the component of y orthogonal to 
z = y - Py_W
print(z)

# computing the magnitude of z to find the distance
dist = np.sqrt(np.dot(z,z))
dist
```

```
[0.90909091 2.72727273 0.90909091]
```

```
3.0151134457776365
```

```
#(d)

#normalizing u1
e1 = u1/ np.sqrt(np.dot(u1,u1))
print('e1 = ', e1, '\n')


#normalizing u2
e2 = u2/ np.sqrt(np.dot(u2,u2))
print('e2 = ', e2, '\n')
```

```
e1 =  [-0.86164044  0.12309149  0.49236596]

e2 =  [-0.40824829  0.40824829 -0.81649658]
```

```
#(e)

# Finding the matrix whose columns are e1,e2

U = np.array([e1,e2]).T
```

(continues on next page)

(continued from previous page)

```
# computing the projection using (*) formula in Theorem 4
U @ U.T @ y
```

```
array([ 0.09090909, -1.72727273,  5.09090909])
```

SECTION 9.6.3 The Gram–Schmidt Process

1. Let $\vec{x}_1 = \begin{bmatrix} 3 \\ 1 \\ -1 \\ 3 \end{bmatrix}, \vec{x}_2 = \begin{bmatrix} -5 \\ 1 \\ 5 \\ -7 \end{bmatrix}, \vec{x}_3 = \begin{bmatrix} 1 \\ 1 \\ -2 \\ 8 \end{bmatrix}$, and $W = \text{span}(\{\vec{x}_1, \vec{x}_2, \vec{x}_3\})$. Construct an orthonormal basis for W.

Solution:

```
x1 = np.array([3,1,-1,3])

x2 = np.array([-5,1,5,-7])

x3 = np.array([1,1,-2,8])

v1 = x1
# projection of x2 onto x1

proj_x2_v1 = (np.dot(x2,v1)/ np.dot(v1,v1))* v1
print( "The projection of x2 onto x1 is ", proj_x2_v1, '\n')

v2 = x2 - proj_x2_v1

print("The component of x1 orthogonal to W1 is", v2,'\n' )

# projection of x3 onto W2

proj_x3_W2 = (np.dot(x3,v1)/ np.dot(v1,v1))* v1  +  (np.dot(x3,v2)/ np.dot(v2,v2))* v2

print( "The projection of x3 onto W2 is ", proj_x3_W2, '\n')

# component of y orthogonal to p
v3 = x3 - proj_x3_W2

print("The component of x3 orthogonal to W2 is", v3 )
```

```
The projection of x2 onto x1 is  [-6. -2.  2. -6.]

The component of x1 orthogonal to W1 is [ 1.  3.  3. -1.]

The projection of x3 onto W2 is  [ 4.  0. -3.  5.]

The component of x3 orthogonal to W2 is [-3.  1.  1.  3.]
```

```
# The unit vector of v1
e1 = v1 / np.sqrt(np.dot(v1,v1))

# The unit vector of v1
e2 = v2 / np.sqrt(np.dot(v2,v2))
```

(continues on next page)

(continued from previous page)

```
# The unit vector of v1
e3 = v3 / np.sqrt(np.dot(v3,v3))

print("e1 = ", e1)
print("e2 = ", e2)
print("e3 = ", e3)
```

```
e1 =  [ 0.67082039  0.2236068  -0.2236068   0.67082039]
e2 =  [ 0.2236068   0.67082039  0.67082039 -0.2236068 ]
e3 =  [-0.67082039  0.2236068   0.2236068   0.67082039]
```

2. Let A be a matrix whose columns are $\vec{x}_1$, $\vec{x}_2$, $\vec{x}_3$ from Exercise 1: $A = \begin{bmatrix} 3 & -5 & 1 \\ 1 & 1 & 1 \\ -1 & 5 & -2 \\ 3 & -7 & 8 \end{bmatrix}$. Find a QR decomposition for A.

Solution:

```
# setup A
A = np.transpose([x1, x2, x3])
print('A = ', A, '\n')

# setup Q
Q = np.transpose([e1, e2, e3])
print('Q = ', Q, '\n')

# Compute R= Q^T*A
R = np.transpose(Q) @ A
print('R = ', R)
```

```
A =  [[ 3 -5  1]
 [ 1  1  1]
 [-1  5 -2]
 [ 3 -7  8]]

Q =  [[ 0.67082039  0.2236068  -0.67082039]
 [ 0.2236068   0.67082039  0.2236068 ]
 [-0.2236068   0.67082039  0.2236068 ]
 [ 0.67082039 -0.2236068   0.67082039]]

R =  [[ 4.47213595e+00 -8.94427191e+00  6.70820393e+00]
 [ 2.77555756e-17  4.47213595e+00 -2.23606798e+00]
 [ 1.11022302e-16  3.33066907e-16  4.47213595e+00]]
```

Let's use `np.linalg.qr` to check our solution:

```
Q, R = np.linalg.qr(A)
print('Q = ', Q, '\n')
print('R = ', R)
```

```
Q =  [[-0.67082039 -0.2236068  -0.67082039]
 [-0.2236068  -0.67082039  0.2236068 ]
 [ 0.2236068  -0.67082039  0.2236068 ]
 [-0.67082039  0.2236068   0.67082039]]
```

(continues on next page)

(continued from previous page)

```
R =  [[-4.47213595  8.94427191 -6.70820393]
 [ 0.         -4.47213595  2.23606798]
 [ 0.          0.          4.47213595]]
```

3. Let $A = QR$, where Q is an $m \times n$ matrix with orthogonal columns, and R is an $n \times n$ matrix. Show that if the columns of A are linearly dependent, then R cannot be invertible.

Solution:

Since the columns of matrix A are linearly dependent, by the invertible matrix theorem, there exists a nonzero vector $\vec{x}$ such that $A\vec{x} = 0$. Replacing A with its QR factorization, we have $QR\vec{x} = 0$. As Q is invertible, we can apply Q^{-1} to both sides, resulting in $R\vec{x} = 0$. Since $\vec{x}$ is nonzero, $R\vec{x} = 0$ has a nontrivial solution. Therefore, by the invertible matrix theorem again, R is not invertible.

SECTION 9.6.4 Least-Squares Problems

1. Let $A = \begin{bmatrix} 1 & -3 & -3 \\ 1 & 5 & 1 \\ 1 & 7 & 2 \end{bmatrix}$ and $\vec{b} = \begin{bmatrix} 5 \\ -3 \\ -5 \end{bmatrix}$.

   a. Find a least-squares solution of $A\vec{x} = \vec{b}$.

   b. Compute the associated least-squares error $\|\vec{b} - A\hat{x}\|$.

```
# (a)

#setup

A = np.array([[1,-3,-3], [1,5,1],[1,7,2]])

b = np.array([[5,-3,-5]]).T

# QR factorization
Q, R = np.linalg.qr(A)

#computing x_hat
x_hat = np.linalg.inv(R) @ Q.T @ b
x_hat
```

```
array([[-1.],
       [ 1.],
       [-4.]])
```

```
#(b)

#compute the least square error
z = b - A @ x_hat
np.linalg.norm(z)
```

```
5.196152422706632
```

2. Describe all least-squares solutions of the system

$x+yx+y=2=4$

Solution:

Let A=[1111] , and $\vec{b}$ =[24]. We find the normal equation of $A\vec{x}$ =$\vec{b}$

$ATA\vec{x} = AT\vec{b}$

```
A = np.array([[1,1],[1,1]])

b = np.array([[2,4]]).T

#compute A^TA

ATA = A.T @ A

print('ATA = \n', ATA,'\n')

#compute A^Tb

ATb = A.T @ b
print('ATb = \n', ATb)
```

```
ATA =
 [[2 2]
 [2 2]]

ATb =
 [[6]
 [6]]
```

Thus, the normal equation is

$$\begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 6 \\ 6 \end{bmatrix}$$

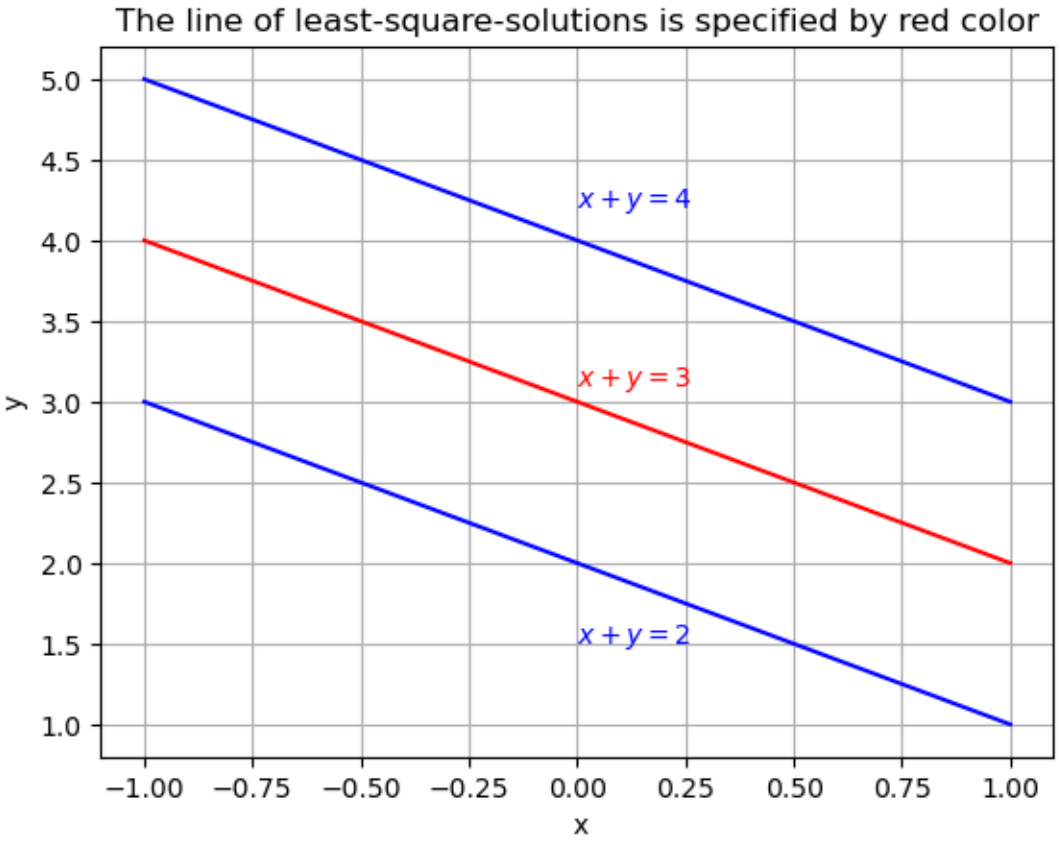
whose solution is the set of all (x, y) such that $x + y = 3$. The solutions form a line midway between $x + y = 2$ and $x + y = 4$.

```
import matplotlib.pyplot as plt

# Data for plotting

x = range(-1, 2)
y1 = [ 2 - k for k in x]
y2 = [ 3 - k for k in x]
y3 = [ 4 - k for k in x]

fig, ax = plt.subplots()

# Specify the length of each axis
#ax.set_xlim(-5,5)
#ax.set_ylim(-5,5)

#Plot x and y1 using blue color
ax.plot(x, y1, color = 'b')
ax.text(0,1.5,'$x + y = 2$', color = 'b')

#Plot x and y2 using red color
ax.plot(x, y2, color = 'r')
ax.text(0,3.1,'$x + y = 3$', color = 'r')

#Plot x and y3 using blue color
ax.plot(x, y3, color = 'b')
```

(continues on next page)

(continued from previous page)

```
ax.text(0,4.2,'$x + y = 4$', color = 'b')


# Add labels and a title to the graph
ax.set(xlabel=' x', ylabel='y', title = 'The line of least-square-solutions is␣
 ↪specified by red color')
ax.grid()

plt.show()
```



9.8 JNB Lab: Linear Algebra

This lab was designed to cement skills the reader has become familiar with throughout the chapter. First, solve by hand. Then, verify your solution using Python Jupyter Notebooks.

9.8.1 Linear Systems

1. Put A into RREF.

$$A = \begin{bmatrix} 1 & 1 & 1 & 2 \\ 2 & 3 & 1 & 3 \\ 1 & -1 & -2 & -11 \end{bmatrix}$$

2. Find the solution set of the following system of equations

$$x - y + 3z = 0$$

$$2x - y + 4z = 1$$

$$-x + 2y - 4z = 2$$

3. Determine if the following vectors form a linearly independent or a linearly dependent set:

a.

$$\vec{v_1} = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} + \begin{bmatrix} 3 \\ 1 \\ 5 \end{bmatrix} + \begin{bmatrix} 4 \\ 2 \\ 3 \end{bmatrix}$$

b.

$$\vec{v_2} = \begin{bmatrix} 1 \\ -4 \\ -2 \\ 0 \end{bmatrix} + \begin{bmatrix} -3 \\ 12 \\ -6 \\ 0 \end{bmatrix}$$

4. Write the general solution of the linear system corresponding to augmented matrix D.

$$A = \begin{bmatrix} 2 & 3 & 0 & 1 \\ 2 & 4 & 2 & 2 \\ 1 & 2 & 1 & 1 \end{bmatrix}$$

9.8.2 Matrices and Determinants

1. Let $A = \begin{bmatrix} 3 & 5 \\ 0 & 2 \\ 1 & 1 \end{bmatrix}$, $B = \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix}$ and $C = \begin{bmatrix} 1 & 0 & 7 \\ 0 & 6 & 0 \\ 0 & 4 & 3 \\ 2 & 0 & 1 \end{bmatrix}$. Find the following matrices, if they exist:

   a. CAB

   b. AB

   c. BCA

   d. B^2

   e. A^2

   f. B^{-1}

   g. C^{-1}

   h. A^tC^t

2. Find the determinant and then the inverse of $A = \begin{bmatrix} 7 & 6 & 54 \\ 0 & 3 & 6 \\ 0 & 0 & 1 \end{bmatrix}$. Prove $det(A^{-1}) = \frac{1}{det(A)}$.

3. Find the volume of the parallelepiped formed by vectors $\begin{bmatrix} 1 \\ 3 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 0 \\ 2 \\ 5 \end{bmatrix}$ and $\begin{bmatrix} 5 \\ 2 \\ 4 \end{bmatrix}$. Are the points at the tips of the vector coplanar? Explain.

9.8.3 Linear Transformations

1. Determine if the following transformations are linear. Explain why or why not.

   a. $T1(\begin{bmatrix} x \\ y \\ z \end{bmatrix}) = \begin{bmatrix} x - 3y - z \\ 3x + 2y - 7z \end{bmatrix}$

   b. $T2(\begin{bmatrix} x \\ y \\ z \end{bmatrix}) = \begin{bmatrix} x + y + z \\ 5z \\ x + 2 \end{bmatrix}$

   c. $T3(\begin{bmatrix} x \\ y \end{bmatrix}) = \begin{bmatrix} x \\ x + y \\ y \end{bmatrix}$

2. Find standard matrices for any linear transformations found in problem 1, if they exist.

3. Verify the Rank-Nullity Theorem for standard matrices found in problem 2, if they exist.

9.8.4 Eigenvalues and Eigenvectors

1. Find the eigenvalues of $A = \begin{bmatrix} 0 & 4 & 8 \\ 0 & 3 & 0 \\ 2 & 7 & 4 \end{bmatrix}$ and their multiplicities.

2. For each eigenvalue in part 1 find the corresponding eigenspace.

3. Find the dimension of each eigenspace in part 2 by finding a basis for it.

9.8.5 Orthogonality

1. Check if $\vec{u}$ and $\vec{v}$ are orthogonal:

   a. $\vec{u} = \begin{bmatrix} 6 \\ 2 \\ 3 \\ 1 \end{bmatrix}$ and $\vec{v} = \begin{bmatrix} 2 \\ -12 \\ 3 \\ 3 \end{bmatrix}$

   b. $\vec{u} = \begin{bmatrix} 7 \\ -3 \\ -2 \end{bmatrix}$ and $\vec{v} = \begin{bmatrix} 2 \\ 7 \\ -3 \end{bmatrix}$

2. Suppose

$\vec{y} = \begin{bmatrix} 7 \\ 9 \end{bmatrix}$ and $\vec{u} = \begin{bmatrix} -1 \\ -2 \end{bmatrix}$.

Compute the $\text{proj}_{\vec{u}}(\vec{y})$ and the component of $\vec{y}$ orthogonal to $\vec{u}$

3. Suppose

$$\vec{u}_1 = \begin{bmatrix} -2 \\ 3 \\ 7 \\ -1 \end{bmatrix},$$

$$\vec{u}_2 = \begin{bmatrix} 2 \\ -2 \\ 0 \\ 4 \end{bmatrix},$$

$W = \text{span}(\vec{u}_1, \vec{u}_2)$ and

$$\vec{y} = \begin{bmatrix} 4 \\ 2 \\ 3 \\ 1 \end{bmatrix}.$$

Compute the $\text{proj}_W(\vec{y})$ and the component of $\vec{y}$ orthogonal to $\vec{u}$.

4. Find a least-squares solution of the inconsistent system $A\vec{x} = \vec{b}$ for

$$A = \begin{bmatrix} 1 & 2 & 4 \\ 1 & -2 & 4 \end{bmatrix} \text{ and } \vec{b} \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

```
import matplotlib.pyplot as plt
import numpy as np
```

9.9 JNB Lab Solutions

This guide focuses on solutions found using Python Jupyter Notebook.

9.9.1 Linear Systems Solutions

1. Put A into RREF.

$$A = \begin{bmatrix} 1 & 1 & 1 & 2 \\ 2 & 3 & 1 & 3 \\ 1 & -1 & -2 & -11 \end{bmatrix}$$

Solution:

```
# Let's set up the matrix before we use row operations
A = np.array([[1,1,1,2],[2,3,1,3],[1,-1,-2,-11]])

A1 = replace(A, 1, 0, -2)
A2 = replace(A1, 2, 0, -1)
A3 = replace(A2, 2, 1, 2)
A4 = scale(A3, 2, -1/5)
A5 = replace(A4, 1, 2, 1)
A6 = replace(A5, 0, 1, -1)
A7 = replace(A6, 0, 2, -1)
A7
```

```
array([[ 1.,  0.,  0., -3.],
       [ 0.,  1.,  0.,  2.],
       [-0., -0.,  1.,  3.]])
```

A7 is in RREF.

2. Find the solution set of the following system of equations

$$x - y + 3z = 0$$

$$6x - y + 4z = 1$$

$$3x + 2y - 4z = 2$$

Solution:

We can change the system of equations into augmented matrix B, and put it into RREF to solve.

```
B = np.array([[1,-1,3,0],[6,-1,4,1],[3,2,-4,2]])

B1 = replace(B, 1, 2, -2)
B2 = replace(B1, 2, 0, -3)
B3 = swap(B2, 1, 2)
B4 = replace(B3, 2, 1, 1)
B5 = replace(B4, 1, 2, 1)
B6 = scale(B5, 2, -1)
B7 = replace(B6, 1, 2, 14)
B8 = scale(B7, 1, 1/5)
B9 = replace(B8, 0, 1, 1)
B10 = replace(B9, 0, 2, -3)
B10
```

```
array([[ 1.,  0.,  0.,  0.],
       [ 0.,  1.,  0.,  3.],
       [-0., -0.,  1.,  1.]])
```

B10 is in reduced echelon form, and gives us the solutions $x = 0$, $y = 3$ and $z = 1$.

3. Determine if the following vectors form a linearly independent or a linearly dependent set:

a.

$$\vec{v_1} = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} + \begin{bmatrix} 3 \\ 1 \\ 5 \end{bmatrix} + \begin{bmatrix} 4 \\ 2 \\ 3 \end{bmatrix}$$

b.

$$\vec{v_2} = \begin{bmatrix} 1 \\ -4 \\ -2 \\ 0 \end{bmatrix} + \begin{bmatrix} -3 \\ 12 \\ -6 \\ 0 \end{bmatrix}$$

Solution:

a. A set of vectors forms a linearly independent set if a matrix (C) made up of the vectors has only the trivial solution. Let's find out the type of solution:

```
C = np.array([[1,3,4,0],[2,1,2,0],[0,5,3,0]])
C1 = replace(C, 1, 0, -2)
C2 = replace(C1, 2, 1, 1)
C2
```

```
array([[ 1.,  3.,  4.,  0.],
       [ 0., -5., -6.,  0.],
       [ 0.,  0., -3.,  0.]])
```

C2 gives us $-3 = 0$, which is impossible and means the only solution is the trivial solution. The set of vectors is linearly independent.

b. By theorem 3, a set of 2 vectors is linearly dependent if and only if one is a multiple of another. The second vector is a multiple of the first by a factor of -3, meaning the set of vectors is linearly dependent.

4. Write the general solution of the linear system corresponding to the augmented matrix D.

$$D = \begin{bmatrix} 2 & 3 & 0 & 1 \\ 2 & 4 & 2 & 2 \\ 1 & 2 & 1 & 1 \end{bmatrix}$$

Solution:

```
D = np.array([[2,3,0,1],[1,2,1,1],[2,4,2,2]])
D1 = replace(D, 0, 1, -1)
D2 = replace(D1, 2, 1, -2)
D3 = replace(D2, 1, 0, -1)
D4 = replace(D3, 0, 1, -1)
D4
```

```
array([[ 1.,  0., -3., -1.],
       [ 0.,  1.,  2.,  1.],
       [ 0.,  0.,  0.,  0.]])
```

Since we are left with an empty row, we see that for any $y \in \mathbb{R}$ we have a unique solution. A general solution for this system is:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{1}{2} - \frac{3}{2}y \\ y \\ \frac{1}{2} - \frac{y}{2} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \end{bmatrix} + \begin{bmatrix} -\frac{3}{2}y \\ y \\ -\frac{y}{2} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \end{bmatrix} + y \begin{bmatrix} -\frac{3}{2} \\ 1 \\ -\frac{1}{2} \end{bmatrix}$$

9.9.2 Matrices and Determinants Solutions

1. Let $A = \begin{bmatrix} 3 & 5 \\ 0 & 2 \\ 1 & 1 \end{bmatrix}$, $B = \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix}$ and $C = \begin{bmatrix} 1 & 0 & 7 \\ 0 & 6 & 0 \\ 0 & 4 & 3 \\ 2 & 0 & 1 \end{bmatrix}$. Find the following matrices, if they exist:

   a. CAB

   b. AB

   c. BCA

   d. B^2

   e. A^2

   f. B^{-1}

   g. C^{-1}

   h. A^tC^t

Solution:

```
#First, set up arrays
A = np.array([[3,5],[0,2],[1,1]])
B = np.array([[2,2],[2,4]])
C = np.array([[1,0,7],[0,6,0],[0,4,3],[2,0,1]])
```

```
# a.
CAB = C @ A @ B
CAB
```

```
array([[44, 68],
       [24, 48],
       [28, 50],
       [36, 58]])
```

```
# b.
AB = A @ B
AB
```

```
array([[16, 26],
       [ 4,  8],
       [ 4,  6]])
```

```
# c. BCA is not possible, as B does not have the same amount of columns as C has rows.
BCA = B @ C @ A
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[13], line 2
      1 # c. BCA is not possible, as B does not have the same amount of columns as␣
 ↪C has rows.
----> 2 BCA = B @ C @ A

ValueError: matmul: Input operand 1 has a mismatch in its core dimension 0, with␣
 ↪gufunc signature (n?,k),(k,m?)->(n?,m?) (size 4 is different from 2)
```

```
# d. B^2 is the same as BB
BB = B @ B
BB
```

```
array([[ 8, 12],
       [12, 20]])
```

```
# e. A^2 is not possible, as A is not a square matrix.
AA = A @ A
AA
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[15], line 2
      1 # e. A^2 is not possible, as A is not a square matrix.
----> 2 AA = A @ A
      3 AA
```

(continues on next page)

(continued from previous page)

```
ValueError: matmul: Input operand 1 has a mismatch in its core dimension 0, with␣
↪gufunc signature (n?,k),(k,m?)->(n?,m?) (size 3 is different from 2)
```

```
# f. First, define an inverse method:

# Compute the inverse
def inverse_matrix(matrix):
    a, b = matrix[0]
    c, d = matrix[1]
    det = a * d - b * c
    if det == 0:
        print(matrix, "is not invertible")
    elif matrix.shape[0] != matrix.shape[1]:
        print(matrix, "is not invertible as it is not a square matrix")
    else:
        D = 1 / det
        inverse = np.zeros((2, 2))
        inverse[0, 0] = D * d
        inverse[0, 1] = -D * b
        inverse[1, 0] = -D * c
        inverse[1, 1] = D * a
        return(inverse)

inverseOfB = inverse_matrix(B)
inverseOfB
```

```
array([[ 1. , -0.5],
       [-0.5,  0.5]])
```

```
# g. A is not invertible, as it is not a square matrix.

inverseOfA = inverse_matrix(A)
inverseOf
```

```
[[3 5]
 [0 2]
 [1 1]] is not invertible as it is not a square matrix
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[17], line 4
      1 # g. A is not invertible, as it is not a square matrix.
      3 inverseOfA = inverse_matrix(A)
----> 4 inverseOf

NameError: name 'inverseOf' is not defined
```

```
# h.
AtCt = A.T @ C.T
AtCt
```

```
array([[10,  0,  3,  7],
       [12, 12, 11, 11]])
```

2. Find the determinant and then the inverse of $A = \begin{bmatrix} 7 & 6 & 54 \\ 0 & 3 & 6 \\ 0 & 0 & 1 \end{bmatrix}$. Prove $det(A^{-1}) = \frac{1}{det(A)}$.

Solution:

```
#First, define the matrix A and a method for finding the determinants of 3x3 matrices:
A = np.array([[7,6,54],[0,3,6],[0,0,1]])

# Computes a 2x2 determinant
def det2(matrix):

    if matrix.shape == (2, 2):
        return matrix[0,0] * matrix[1,1] - matrix[0,1] * matrix[1,0]
    else:
        print("Please enter a 2x2 matrix")
# Computes the 3x3 determinant over a given row
def det3(matrix, row):
    if matrix.shape == (3, 3) and row in range(3):
        det = 0
        for j in range(3):
            submatrix = np.delete( (np.delete(matrix, row , 0)), j ,1)
            det += (-1) ** (row + j) * matrix[row, j] * det2(submatrix)
        return det
    else:
        print("The input matrix is not 3x3 or the row is not chosen in the right range
 ↪")

#Calculate the determinant of A
detA = det3(A,2)
detA
```

```
21
```

```
#Calculate the inverse of A using numpy
invA = np.linalg.inv(A)
invA
```

```
array([[ 0.14285714, -0.28571429, -6.        ],
       [ 0.        ,  0.33333333, -2.        ],
       [ 0.        ,  0.        ,  1.        ]])
```

```
#Calculate the determinant of the inverse of A
detInvA = det3(invA,0)
detInvA
```

```
0.047619047619047616
```

```
#Finally, we can use a simple boolean to prove det(A^-1) = 1/det(A)
inverseCheck = (detInvA == 1/detA)
print('Is the determinant of the inverse equal to the inverse of the determinant of A?
 ↪:', inverseCheck)
```

```
Is the determinant of the inverse equal to the inverse of the determinant of A?:␣
 ↪True
```

3. Find the volume of the parallelepiped formed by vectors $\begin{bmatrix}1\\3\\0\end{bmatrix}$, $\begin{bmatrix}0\\2\\5\end{bmatrix}$ and $\begin{bmatrix}5\\2\\4\end{bmatrix}$. Are the points at the tips of the vector coplanar? Explain.

Solution:

```
# We can form a matrix P out of the vectors in which we can find the determinant. The
 ↪determinant is equal to the volume.
P = np.array([[1,0,5],[3,2,2],[0,5,4]])

volume = det3(P,2) #the 'det3' function was defined in question 2
print ('volume =', volume)
```

```
volume = 73
```

The vector tips cannot be coplanar, as the determinant is not equal to zero. Conceptually, if there is a volume above zero, then the vectors could not lie on the same plane.

9.9.3 Linear Transformations Solutions

1. Determine if the following transformations are linear. Explain why or why not.

   a. $T1(\begin{bmatrix}x\\y\\z\end{bmatrix}) = \begin{bmatrix}x-3y-z\\3x+2y-7z\end{bmatrix}$

   b. $T2(\begin{bmatrix}x\\y\\z\end{bmatrix}) = \begin{bmatrix}x+y+z\\5z\\x+2\end{bmatrix}$

   c. $T3(\begin{bmatrix}x\\y\end{bmatrix}) = \begin{bmatrix}x\\x+y\\y\end{bmatrix}$

Solution:

For each transformation, we will check the two conditions of a linear transformation:

1. $T(\vec{0}) = \vec{0}$
2. $T(c\vec{u} + d\vec{v}) = cT(\vec{u}) + dT(\vec{v}) \quad \forall \vec{u}, \vec{v} \in \mathbb{R}^m \quad \text{and} \quad \forall c \in \mathbb{R}$

These conditions can be checked using Python functions.

```
#a

# transformation function for T1
def T1(V):
    W = np.zeros((2,1)) #returns the zero vector in R^2
    W[0,0] = V[0,0] - 3*V[1,0] - V[2,0]   #builds first component
    W[1,0] = 3*V[0,0] + 2*V[1,0] - 7*V[2,0]   #builds second component
    return W

Z = np.zeros((3,1))

T1(Z)
```

```
array([[0.],
       [0.]])
```

```
# The first condition is fulfilled, and we can test the second.

c = 2
d = 4
u = np.array([[-1],[5],[3]])
v = np.array([[3],[-8],[6]])

T1(c * u + d * v)
```

```
array([[  46.],
       [-224.]])
```

```
c * T1(u) + d * T1(v)
```

```
array([[  46.],
       [-224.]])
```

Both conditions are true, so transformation T1 is linear.

Important Note: Normally, the surefire way to prove the second condition would be with arbitrary values. However, due to how Python works we need to use literal values instead of arbitrary values. This will not be mentioned in further examples, but we can for sure prove $T1(c\vec{u} + d\vec{v}) = cT1(\vec{u}) + dT1(\vec{v})$ with the following proof:

Suppose $\vec{u} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \vec{v} = \begin{bmatrix} e \\ f \\ g \end{bmatrix}$ and $\quad c, d, x, y, z, e, f, g \in \mathbb{R}$

$$T1(c \begin{bmatrix} x \\ y \\ z \end{bmatrix} + d \begin{bmatrix} e \\ f \\ g \end{bmatrix}) = T(\begin{bmatrix} cx \\ cy \\ cz \end{bmatrix} + \begin{bmatrix} de \\ df \\ dg \end{bmatrix}) = T(\begin{bmatrix} cx + de \\ cy + df \\ cz + dg \end{bmatrix} = \begin{bmatrix} cx + de - 3(cy + df) - (cz + dg) \\ 3(cx + de) + 2(cy + df) - 7(cz + dg) \end{bmatrix}$$

$$cT1(\begin{bmatrix} x \\ y \\ z \end{bmatrix}) + dT1(\begin{bmatrix} e \\ f \\ g \end{bmatrix}) = c \begin{bmatrix} x - 3y - z \\ 3x + 2y - 7z \end{bmatrix} + d \begin{bmatrix} e - 3f - g \\ 3e + 2f - 7g \end{bmatrix} = \begin{bmatrix} cx - c3y - cz \\ c3x + c2y - c7z \end{bmatrix} + \begin{bmatrix} de - d3f - dg \\ d3e + d2f - d7g \end{bmatrix} = \begin{bmatrix} cx + de - 3(c \\ 3(cx + de) + 2(\end{bmatrix}$$

Therefore, $T1(c\vec{u} + d\vec{v}) = cT1(\vec{u}) + dT1(\vec{v})$.

```
# b

# transformation function for T2
def T2(V):
    W = np.zeros((3,1)) #returns the zero vector in R^3
    W[0,0] = V[0,0] + V[1,0] + V[2,0]   #builds first component
    W[1,0] = 5*V[2,0]   #builds second component
    W[1,0] = V[0,0] + 2   #builds third component
    return W

Z = np.zeros((3,1))

T2(Z)
```

```
array([[0.],
       [2.],
       [0.]])
```

The first condition is not upheld, so T2 cannot be a linear transformation.

```
# c

# transformation function for T3
def T3(V):
    W = np.zeros((3,1)) #returns the zero vector in R^3
    W[0,0] = V[0,0]   #builds first component
    W[1,0] = V[0,0] + V[1,0]   #builds second component
    W[2,0] = V[1,0]   #builds third component
    return W

Z = np.zeros((2,1))

T3(Z)
```

```
array([[0.],
       [0.],
       [0.]])
```

```
# The first condition is fulfilled, and we can test the second.

c = 3
d = 5
u = np.array([[-2],[11]])
v = np.array([[5],[-3]])

T3(c * u + d * v)
```

```
array([[19.],
       [37.],
       [18.]])
```

```
c * T3(u) + d * T3(v)
```

```
array([[19.],
       [37.],
       [18.]])
```

Both conditions are upheld for T3, meaning it is a linear transformation.

T1 and T3 are linear transformations, while T2 is not.

2. Find standard matrices for any linear transformations found in problem 1, if they exist.

Solution:

a. Standard matrix for T1: $A = \begin{bmatrix} 1 & -3 & -1 \\ 3 & 2 & -7 \end{bmatrix}$

b. T2 is not a linear transformation, so there can be no standard matrix.

c. Standard matrix for T3: $A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}$

We can prove these are the standard matrices by using Python:

```
# define the standard matrices
STD_T1 = np.array([[1, -3, -1],[3, 2, -7]])
STD_T3 = np.array([[1, 0],[1, 1],[0, 1]])

# define some test vectors to transform
U_T1 = np.array([[2],[-5],[4]])
U_T3 = np.array([[-3],[6]])

# store the vectors transformed by their respective functions
func_T1_test = T1(U_T1)
func_T3_test = T3(U_T3)

# store the vectors transformed by their respective standard matrices
STD_T1_test = STD_T1 @ U_T1
STD_T3_test = STD_T3 @ U_T3

# print the results using booleans to confirm the standard matrices are correct
print('STD_1 is the standard matrix for T1 = ', STD_T1_test == func_T1_test)
print('STD_1 is the standard matrix for T1 = ', STD_T3_test == func_T3_test)
```

```
STD_1 is the standard matrix for T1 =  [[ True]
 [ True]]
STD_1 is the standard matrix for T1 =  [[ True]
 [ True]
 [ True]]
```

Both standard matrices are correct.

3. Verify the Rank-Nullity Theorem for standard matrices found in problem 2, if they exist.

Solution:

From the Rank-Nullity theorem, we know $dim(col(A))$ = the number of pivot columns in A and $dim(null(A))$ = the number of non-pivot columns in A. To find these, we will need to put the standard matrices into RREF.

```
# a

STD_T1 = np.array([[1, -3, -1],[3, 2, -7]])
STD_T1_1 = replace(STD_T1, 1, 0, -3)
STD_T1_2 = scale(STD_T1_1, 1, 1/11)
STD_T1_3 = replace(STD_T1_2, 0, 1, 3)
STD_T1_3
```

```
array([[ 1.        ,  0.        , -2.09090909],
       [ 0.        ,  1.        , -0.36363636]])
```

Column 1 and 2 are pivots columns, while the third column is not. Therefore, $dim(col(A)) + dim(null(A)) = 2 + 1 = 3 = n$.

```
# c

STD_T3 = np.array([[1, 0],[1, 1],[0, 1]])
STD_T3_1 = replace(STD_T3, 1, 0, -1)
STD_T3_2 = replace(STD_T3_1, 2, 1, -1)
STD_T3_2
```

```
array([[1., 0.],
       [0., 1.],
       [0., 0.]])
```

Column 1 and 2 are pivots columns. Therefore, $dim(col(A)) + dim(null(A)) = 2 + 0 = 2 = n$.

9.9.4 Eigenvalues and Eigenvectors Solutions

1. Find the eigenvalues of $A = \begin{bmatrix} 0 & 4 & 8 \\ 0 & 3 & 0 \\ 2 & 7 & 4 \end{bmatrix}$ and their multiplicities.

Solution:

```
#define the vector
A = np.array([[0,4,8], [0, 3, 0], [2,7,4]])

#eigenvalues of A
eigenvalues = np.linalg.eigvals(A)
eigenvalues
```

```
array([-2.47213595,  6.47213595,  3.        ])
```

The multiplicity of each eigenvalue is 1, as they only appear once in the array.

2. For each eigenvalue in part 1 find the corresponding eigenspace.

Solution:

$$E_{-2.47213595} = Null(\begin{bmatrix} 0+2.47213595 & 4 & 8 \\ 0 & 3+2.47213595 & 0 \\ 2 & 7 & 4+2.47213595 \end{bmatrix}) = Null(\begin{bmatrix} 2.47213595 & 4 & 8 \\ 0 & 5.47213595 & 0 \\ 2 & 7 & 6.47213595 \end{bmatrix})$$

$$E_{6.47213595} = Null(\begin{bmatrix} 0-6.47213595 & 4 & 8 \\ 0 & 3-6.47213595 & 0 \\ 2 & 7 & 4-6.47213595 \end{bmatrix}) = Null(\begin{bmatrix} -6.47213595 & 4 & 8 \\ 0 & -3.47213595 & 0 \\ 2 & 7 & -2.47213595 \end{bmatrix})$$

$$E_3 = Null(\begin{bmatrix} 0-3 & 4 & 8 \\ 0 & 3-3 & 0 \\ 2 & 7 & 4-3 \end{bmatrix}) = Null(\begin{bmatrix} -3 & 4 & 8 \\ 0 & 0 & 0 \\ 2 & 7 & 1 \end{bmatrix})$$

3. Find the dimension of each eigenspace in part 2 by finding a basis for it.

Solution:

Basis for $E_{-2.47213595}$: There is no basis, as the null space is just the zero vector.

Basis for $E_{6.47213595}$: There is no basis, as the null space is just the zero vector.

Basis for E_3 : $\left\{ \begin{bmatrix} 52/29 \\ -19/29 \\ 1 \end{bmatrix} \right\}$

9.9.5 Orthogonality Solutions

1. Check if $\vec{u}$ and $\vec{v}$ are orthogonal:

   a. $\vec{u} = \begin{bmatrix} 6 \\ 2 \\ 3 \\ 1 \end{bmatrix}$ and $\vec{v} = \begin{bmatrix} 2 \\ -12 \\ 3 \\ 3 \end{bmatrix}$

   b. $\vec{u} = \begin{bmatrix} 7 \\ -3 \\ -2 \end{bmatrix}$ and $\vec{v} = \begin{bmatrix} 2 \\ 7 \\ -3 \end{bmatrix}$

Solution:

```
# a
u = np.array([6, 2, 3, 1]) #In this case, we use standard arrays to conform to how np.
 ↪dot is defined
v = np.array([2, -12, 3, 3])
uv = np.dot(u,v)
uv
```

```
0
```

The dot product of $\vec{u}$ and $\vec{v}$ is 0, so $\vec{u}$ and $\vec{v}$ are orthogonal.

```
# b
u = np.array([7, -3, -2]) #In this case, we use standard arrays to conform to how np.
 ↪dot is defined
v = np.array([2, 7, -3])
uv = np.dot(u,v)
uv
```

```
-1
```

The dot product of $\vec{u}$ and $\vec{v}$ is NOT 0, so $\vec{u}$ and $\vec{v}$ are NOT orthogonal.

2. Suppose

$\vec{y} = \begin{bmatrix} 7 \\ 9 \end{bmatrix}$ and $\vec{u} = \begin{bmatrix} -1 \\ -2 \end{bmatrix}$.

Compute the $\text{proj}_{\vec{u}}(\vec{y})$ and the component of $\vec{y}$ orthogonal to $\vec{u}$

Solution:

```
# setup vectors

y = np.array([7,9])

u = np.array([-1,-2])

# projection of y onto u

proj_y_u = (np.dot(y,u)/ np.dot(u,u))* u
print( "the projection of y onto u is ", proj_y_u)

z = y - proj_y_u
```

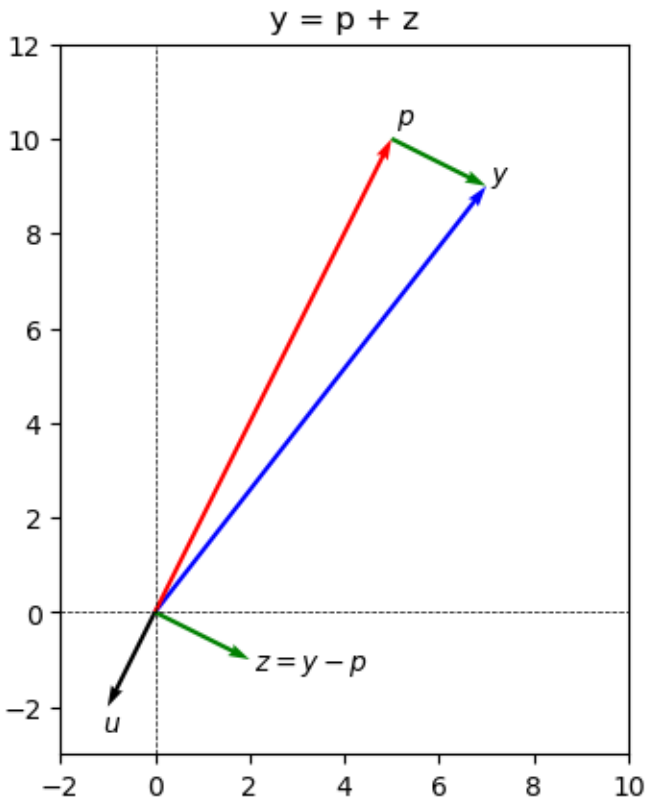
(continues on next page)

(continued from previous page)

```
print("the component of y orthogonal to u is", z )
```

```
the projection of y onto u is  [ 5. 10.]
the component of y orthogonal to u is [ 2. -1.]
```

Now we can plot $\text{proj}_{\vec{u}}(\vec{y})$ and the component of y orthogonal to u in the xy-plane.



3. Suppose

$$\vec{u}_1 = \begin{bmatrix} -2 \\ 3 \\ 7 \\ -1 \end{bmatrix},$$

$$\vec{u}_2 = \begin{bmatrix} 2 \\ -2 \\ 0 \\ 4 \end{bmatrix},$$

$W = \text{span}(\vec{u}_1, \vec{u}_2)$ and

$$\vec{y} = \begin{bmatrix} 4 \\ 2 \\ 3 \\ 1 \end{bmatrix}.$$

Compute the $\text{proj}_W(\vec{y})$ and the component of $\vec{y}$ orthogonal to $\vec{u}$.

Solution:

```
# setup vectors

u1 = np.array([-2,3,7,-1])

u2 = np.array([2,-2,0,4])

y = np.array([3,2,3,1])

# projection of y onto W

proj_y_W = (np.dot(y,u1)/ np.dot(u1,u1))* u1  +  (np.dot(y,u2)/ np.dot(u2,u2))* u2

print( "The projection of y onto W is ", proj_y_W, '\n')

# component of y orthogonal to p
z = y - proj_y_W

print("The component of y orthogonal to u is", z )
```

```
The projection of y onto W is  [-0.13492063  0.45238095  2.22222222  0.68253968]

The component of y orthogonal to u is [3.13492063 1.54761905 0.77777778 0.31746032]
```

4. Find a least-squares solution of the inconsistent system $A\vec{x} = \vec{b}$ for

$$A = \begin{bmatrix} 1 & 2 & 4 \\ 1 & -2 & 4 \end{bmatrix} \text{ and } \vec{b} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

Solution:

```
#First, set up a normal equation for Ax=b and find a solution for it

A = np.array([[1,2,4], [1,-2,4]])

b = np.array([[3], [4]])

# compute A^TA
ATA = A.transpose() @ A

print("A^TA = \n", ATA)

# compute A^Tb
ATb = A.transpose() @ b

print("\n A^Tb = \n", ATb)
```

```
A^TA =
 [[ 2  0  8]
 [ 0  8  0]
 [ 8  0 32]]

 A^Tb =
 [[ 7]
 [-2]
 [28]]
```

Thus, the normal equation is

$$\begin{bmatrix} 2 & 0 & 8 \\ 0 & 8 & 0 \\ 8 & 0 & 32 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ -2 \\ 28 \end{bmatrix}$$

To solve this equation we form its augmented and use row operationsmatrix:

```
#augmented
M = np.concatenate((ATA, ATb), axis=1)
M
```

```
array([[ 2,  0,  8,  7],
       [ 0,  8,  0, -2],
       [ 8,  0, 32, 28]])
```

```
M1 = replace(M, 2, 0, -4)
M2 = scale(M1, 1, 1/8)
M3 = scale(M2, 0, 1/2)
M3
```

```
array([[ 1.  ,  0.  ,  4.  ,  3.5 ],
       [ 0.  ,  1.  ,  0.  , -0.25],
       [ 0.  ,  0.  ,  0.  ,  0.  ]])
```

The general solution is $x_1 = \frac{7}{2} - 4x_3$, $x_2 = -\frac{1}{4}$, and x_3 is a free parameter. The general least squares solution of $A\vec{x} = \vec{b}$ has the form:

$$\hat{x} = \begin{bmatrix} \frac{7}{2} \\ -\frac{1}{4} \\ 0 \end{bmatrix} + x_3 \begin{bmatrix} -4 \\ 0 \\ 1 \end{bmatrix}$$

CHAPTER 10

LINEAR ALGEBRA AND OPTIMIZATION FOR DATA ANALYSIS - Wheaton College Team

Note

In this chapter, applications of linear algebra and optimization to data analysis will be presented by a team from Wheaton College (IL): Dr. Paul Isihara (Project Leader) Luke Altorfer, Olenka Ashwill, Adam Erickson, Audrey Fitzekam, James Heryanto, Josiah Hsu, Jacob Schauer, Carson Slater, Jacob Snyder, Tyler Weiner, and Olivia Wertje (Fall 2022 Math 385 Class)

This material was published previously as the UMAP Module titled Linear Algebra and Optimization in Data Analysis. The UMAP Journal 44 (2): 117–156. Copyright 2023 by COMAP, Inc. All rights reserved. Author: Paul Isihara and Student Team (Luke Altorfer, Olenka Ashwill, AdamErickson, Audrey Fitzekam, James Heryanto, Josiah Hsu, Jacob Schauer, Carson Slater, Jacob Snyder, Tyler Weiner, and Olivia Wertjes. Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice. Abstracting with credit is permitted, but copyrights for components of this work owned by others than COMAP must be honored. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior permission from COMAP.

10.1 Introduction

This chapter offers a brief introduction to several important topics in data analysis:

- Linear approximation of data by OLS regression (Section 2)
- Clustering of unlabelled data using k -means (Section 3)
- Dimensionality reduction of data by principal component analysis (Section 4)
- Binary classification of labeled data using support vector machines (Section 5)

For each topic, we first illustrate the goal or a key idea(s) underlying the data analysis using a real data set (e.g., COVID test rates and death rates by zip code). We then explain how linear algebra and the solution to an optimization problem yield the desired goal. For simplicity, we explain concepts and/or give examples in two-dimensional settings that are easy

to visualize. Aggarwal [2020] gives an in-depth treatment of how the basic theory covered in this module can be extended to more advanced settings.

The exercises include problems solvable by hand to test the reader's understanding of the theory, as well as problems that utilize Python Jupyter Notebooks (JNBs). Instructors can tailor these JNBs for use as computer labs depending on the level of students' familiarity with Python programming. The JNB exercises should at least be read, if not tackled until completion, since they suggest how the basic examples and concepts introduced using small datasets in two dimensions can be applied to real data. The table below indicates the Python Jupyter Notebooks (JNBs) utilized in this chapter. The JNBs are available at https://tinyurl.com/2neb4z4c

Name of JNB	Exercise	Data File
OLS LINEAR REGRESSION	Exercise 2.3	ACTCollegeEligible.csv
K-MEANS CLUSTERING	Exercise 3.3	CPDdistricts.geojson
PRINCIPAL COMPONENT ANALYSIS	Exercises 4.5, 4.6	standardizedindicators.xlsx
SUPPORT VECTOR MACHINES	Exercise 5.3	Housing.xlsx

10.2 OLS Linear Regression

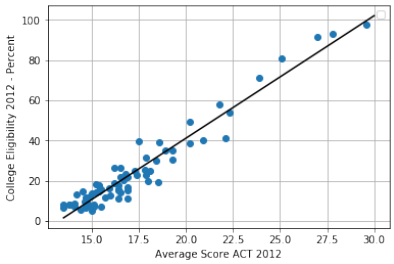
We begin with the familiar topic of ordinary least squares (OLS) linear regression. The table below shows an excerpt of Chicago Public School data for 2011–2012 from the Chicago Data Portal. One expects a higher average ACT score to be associated with a higher percentage of college eligibility.

School zipcode	Average ACT Score	College Eligibility
60605	25.1	80.7
60607	27	91.6
…	…	…
60660	16.5	14.2

Source: Chicago Data Portal,

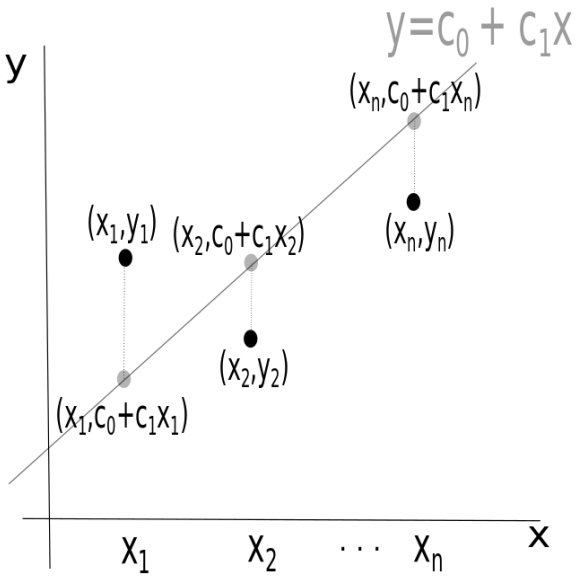
https://data.cityofchicago.org/Education/Chicago-Public-Schools-Progress-Report-Cards-2011-/9xs2-f89t

The figure below shows a scatterplot of $n = 83$ data points partly reproduced in the above table together with the OLS regression line.



10.2.1 Least-Squares Solutions

Consider a collection of n data points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ in $\mathbb{R}^2$. We seek the line $y = c_0 + c_1 x$ that best fits these data, where c_0 is the y-intercept of the line and c_1 the slope of the line. As shown in the figure below, an OLS loss function $J(c_0, c_1)$ sums the squared vertical separations between the data points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ and the line $y = c_0 + c_1 x$.



If the n data points are collinear, then there exists an exact solution (c_0, c_1) to the system

$$c_0 + c_1 x_1 = y_1,$$

$$c_0 + c_1 x_2 = y_2,$$

$$\vdots$$

$$c_0 + c_1 x_n = y_n.$$

In matrix form, this system of equations becomes

$$\mathbf{Ac} = \mathbf{y},$$

where

$$\mathbf{A} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}, \quad \mathbf{c} = \begin{pmatrix} c_0 \\ c_1 \end{pmatrix}, \quad \textit{and} \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}.$$

In general, this linear system with $n > 2$ equations and 2 unknowns (c_0, c_1) will not have an exact solution. Instead, our goal to find a linear fit to the data is based on a *least-squares solution*, one that solves the following optimization problem:

10.2.2 OLS LINEAR REGRESSION OPTIMIZATION PROBLEM

OLS Linear Regression Optimization Problem

Find (c_0, c_1) which minimizes the loss function $J(c_0, c_1)$ defined as

$$J(c_0, c_1) = \frac{1}{2}\sum_{i=1}^{n}(y_i - (c_0 + c_1 x_i))^2 = \frac{1}{2}\|\mathbf{y} - \mathbf{Ac}\|^2.$$

(The factor of 1/2 multiplying the sum is introduced to simplify the theoretical analysis of the loss function.)

The loss function J is zero when the points are collinear and situated on the line $y = c_0 + c_1 x$; otherwise, J is positive, since it is half the sum of the squared vertical separations between data points and the line $y = c_0 + c_1 x$, as shown in the previous figure.

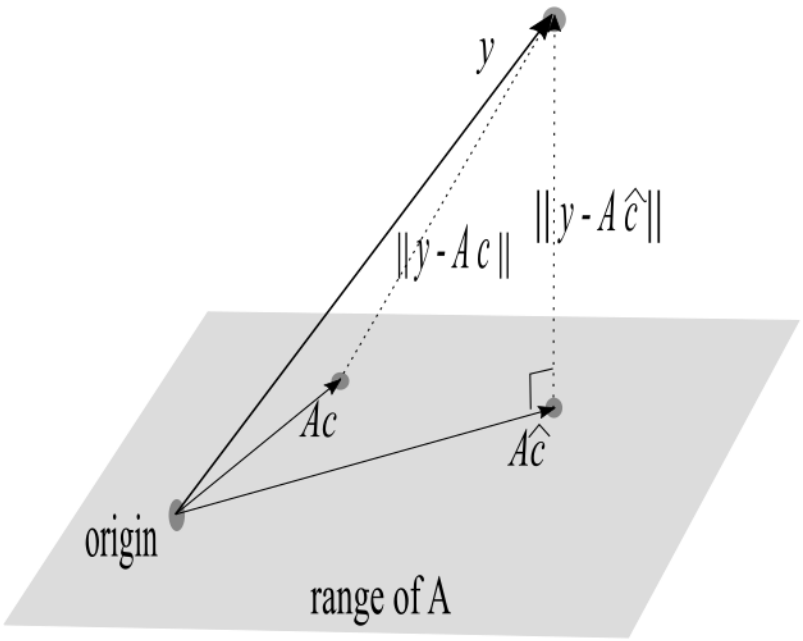
10.2.3 Minimizing the OLS Loss Function via Normal Equations

Let $\mathbf{A}$ be an $n \times 2$ matrix of real numbers, and let the linear transformation $\mathbf{L} : \mathbb{R}^2 \to \mathbb{R}^n$ be defined by $\mathbf{L}(\mathbf{c}) = \mathbf{Ac}$, where $\mathbf{c}$ is a 2×1 column vector. The range of $\mathbf{A}$ is the set of all $n \times 1$ vectors defined by

$$range\,of\,\mathbf{A} = \{\mathbf{Ac} \mid \mathbf{c} \in \mathbf{R}^2\}.$$

By choosing $\mathbf{c} = (1, 0)^T$, we see that the first column of $\mathbf{A}$ (denoted $\mathbf{a}_1$) is in the range of $\mathbf{A}$. By choosing $\mathbf{c} = (0, 1)^T$, we see that the second column of $\mathbf{A}$ (denoted $\mathbf{a}_2$) is also in the range of $\mathbf{A}$.

Minimizing the loss function $J(\mathbf{c}) = \frac{1}{2}\|\mathbf{y} - \mathbf{Ac}\|^2$ is equivalent to minimizing $\|\mathbf{y} - \mathbf{Ac}\|$, the latter being the distance between $\mathbf{y}$ and an arbitrary vector $\mathbf{Ac}$ that is in the range of $\mathbf{A}$ (see figure below). If there is no exact solution (i.e., $\mathbf{y}$ is not in the range of $\mathbf{A}$), this minimization is accomplished by choosing $\hat{\mathbf{c}}$ such that vector $\mathbf{y} - \mathbf{A}\hat{c}$ is orthogonal to the range of $\mathbf{A}$. That is, for $\hat{\mathbf{c}}$ to be a least-squares solution to $\mathbf{Ac} = \mathbf{y}$, the vector $\mathbf{y} - \mathbf{A}\hat{\mathbf{c}}$ must be orthogonal (perpendicular) to each vector in the range of $\mathbf{A}$. Since both column vectors of $\mathbf{A}$ (namely, $\mathbf{a}_1$ and $\mathbf{a}_2$) are in the range of $\mathbf{A}$, it follows that $\mathbf{a}_i \cdot (\mathbf{y} - \mathbf{A}\hat{\mathbf{c}}) = \mathbf{a}_i^T(\mathbf{y} - \mathbf{A}\hat{\mathbf{c}}) = 0$ for each column vector $\mathbf{a}_i$ of matrix~$\mathbf{A}$.



Hence,

$$\mathbf{A}^T(\mathbf{y} - \mathbf{A}\hat{\mathbf{c}}) = \mathbf{0} \Rightarrow$$

$$\mathbf{A}^T\mathbf{y} - \mathbf{A}^T\mathbf{A}\hat{\mathbf{c}} = \mathbf{0} \Rightarrow$$

$$\mathbf{A}^T\mathbf{y} = \mathbf{A}^T\mathbf{A}\hat{\mathbf{c}}$$

The last equation is called the **normal equation** for the system $\mathbf{Ac} = \mathbf{y}$. Solving the normal equations, one obtains optimal values for c_0, c_1.

Example 2.1.

Example 2.1

Consider the data $(-1, 1)$, $(0, 0)$, $(1, 2)$, $(2, 3)$. Use the normal equations to find the OLS regression line for the data.

Solution.

The system is

$$\begin{aligned} c_0 - c_1 &= 1, \\ c_0 &= 0, \\ c_0 + c_1 &= 2, \\ c_0 + 2c_1 &= 3; \end{aligned}$$

or, in matrix form,

$$\begin{pmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 2 \\ 3 \end{pmatrix}.$$

The normal equations are

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \end{pmatrix} \begin{pmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \end{pmatrix}, \tag{10.1}$$

or

$$\begin{pmatrix} 6 \\ 7 \end{pmatrix} = \begin{pmatrix} 4 & 2 \\ 2 & 6 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \end{pmatrix}.$$

The least-squares solution is therefore $c_0 = 11/10$, $c_1 = 4/5$.

10.2.4 Equivalence of Gradient-Based Optimization

Note that the normal equations are equivalent to gradient-based minimization of the OLS linear regression loss function $J(c_0, c_1) = \frac{1}{2}\sum_{i=1}^{n}(y_i - (c_0 + c_1 x_i))^2$:

$$\mathbf{A}^T(\mathbf{y} - \mathbf{A}\hat{\mathbf{c}}) = \begin{pmatrix} 1 & 1 & \dots & 1 \\ x_1 & x_2 & \dots & x_n \end{pmatrix} \begin{pmatrix} y_1 - (c_0 + c_1 x_1) \\ y_2 - (c_0 + c_1 x_2) \\ \dots \\ y_n - (c_0 + c_1 x_n) \end{pmatrix}$$
$$= \begin{pmatrix} \sum_{i=1}^{n}(y_i - (c_0 + c_1 x_i)) \\ \sum_{i=1}^{n} x_i(y_i - (c_0 + c_1 x_i)) \end{pmatrix} = \begin{pmatrix} -\frac{\partial J}{\partial c_0} \\ -\frac{\partial J}{\partial c_1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

The normal equations are equivalent to setting both partial derivatives of J equal to zero, as is required to minimize the loss function $J(c_0, c_1)$.

Example 2.2.

Example 2.2

For the data points in Example 2.1, show how gradient-based optimization of the loss function J gives the same values for c_0 and c_1.

Solution:

The loss function $J(c_0, c_1)$ is

$$J(c_0, c_1) = \tfrac{1}{2}[(1 - (c_0 - c_1))^2 + (0 - c_0)^2 + (2 - (c_0 + c_1))^2 + (3 - (c_0 + 2c_1))^2].$$

To minimize J, we solve the linear system for $\nabla J(c_1, c_2) = \mathbf{0}$:

$$\frac{\partial J}{\partial c_0} = 4c_0 + 2c_1 - 6 = 0, \qquad \frac{\partial J}{\partial c_1} = 2c_0 + 6c_1 - 7 = 0.$$

Solving this system, we obtain $c_0 = 11/10$ and $c_1 = 4/5$.

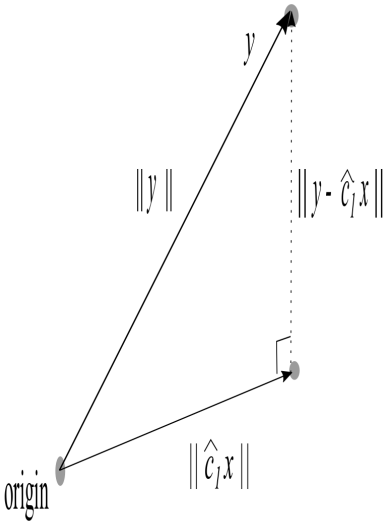
The system used to find these critical values is equivalent to:

$$\begin{pmatrix} -\frac{\partial J}{\partial c_0} \\ -\frac{\partial J}{\partial c_1} \end{pmatrix} = \begin{pmatrix} (1-(c_0-c_1))+(0-c_0)+(2-(c_0+c_1))+(3-(c_0+2c_1)) \\ -(1-(c_0-c_1))+0(0-c_0)+(2-(c_0+c_1))+2(3-(c_0+2c_1)) \end{pmatrix}$$
$$= \begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 2 \\ 3 \end{pmatrix} - \begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \end{pmatrix} \begin{pmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \tag{10.2}$$

Gradient-based optimization expressed is indeed equivalent to the normal equations.

To say that solving the normal equations is mathematically equivalent to gradient-based optimization of the OLS loss function does not imply that the normal equations offer the best numerical method for optimization of the loss function [Epperly 2022]. Beyond the scope of this Module is an assessment of different numerical approaches such as gradient descent and its variants including the Adam method [Sun et. al. 2019], matrix factorizations such as SVD or QR [Aggarwal 2020], and mean-centering and standardizing data [Toth 2020].

In addition to facilitating numerical analysis, mean-centering simplifies the linear algebra approach. Mean-centered data is obtained from a collection of data points by replacing the original dataset (x_i, y_i) with the data points $(x_i - \bar{x}, y_i - \bar{y})$, $i = 1, ... n$, where $\bar{x}$ and $\bar{y}$ are the respective means of the x and y coordinates of the original data. One can show that for mean-centered data, $\hat{c}_0 = 0$, and the regression line is $y = \hat{c}_1 x$. Let $\mathbf{x} = (x_1, x_2, ..., x_n)^{tr}$. For mean-centered data, the regression vector $\hat{\mathbf{y}} = \hat{c}_1 \mathbf{x}$ is obtained by projecting the vector $\mathbf{y}$ directly onto the vector $\mathbf{x}$ as shown in the figure below.



The *variation equation* states that the total variation in the y-values is the sum of the explained variation (variation in the corresponding y-values on the regression line) and the unexplained variation (the sum of squared residuals or twice the value of the loss function J). For mean-centered data, the total variation is $|| \mathbf{y} ||^2$, the explained variation is $|| \hat{c}_1 \mathbf{x} ||^2$ and the unexplained variation is $|| \mathbf{y} - \hat{c}_1 \mathbf{x} ||^2$. In other words, the variation equation for mean-centered data is simply the Pythagorean theorem.

10.2.5 Exercises

Exercises

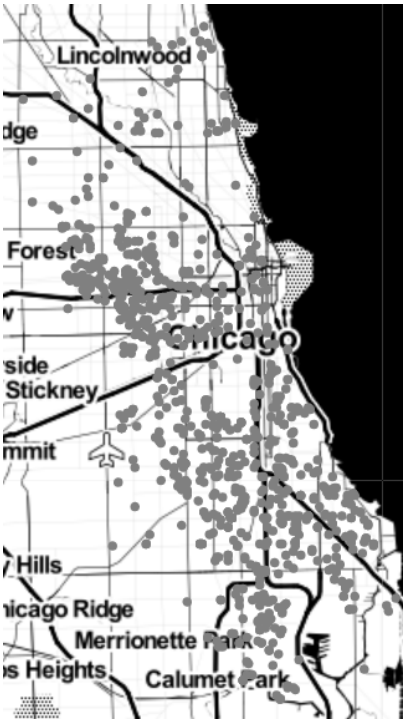
1) Consider the data $(1, 0)$, $(4, 5)$, $(7, 8)$. Use the normal equations to find the least-squares solution line $y = a + bx$ that best fits the data.

2) Consider the data $(-1, 1)$, $(0, 0)$, $(1, 2)$, $(2, 3)$. Use the normal equations to find the least-squares solution for the parabola $y = a + bx + cx^2$ that best fits the data.

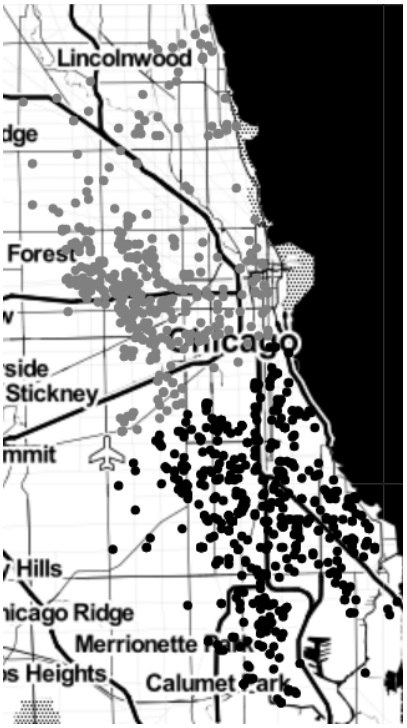
10.3 K-means Clustering

10.3.1 Clustering of Data

In this section, we will consider the problem of clustering into k groups a set of unlabelled points in a coordinate plane $\mathbb{R}^2$. For example, the figure below is a scatterplot of a data set from the Chicago Data Portal that shows the locations of 1,000 recent acts of violence in the city of Chicago. Police departments may use data like this to help identify clusters or "hot spots" in their violence reduction efforts. Another question is whether these crime locations might be naturally divided in some way, say, into two halves.



We can use k-means clustering to separate the mapped crime data as points in $\mathbb{R}^2$ into an arbitrary number of groups solely based on location, as seen in the figure below. This figure shows $k = 2$ clusters. The I-55 Stevenson Expressway, which runs northeast from Stickney to downtown Chicago, is close to the boundary between the two clusters. High-density areas exist within each cluster, one in the West side and the other in much of the South side of the city. These areas reflect a long history of injustices in these parts of Chicago [Amdat 2021].



Data source: Chicago Data Portal https://data.cityofchicago.org/Public-Safety/Violence-Reduction-Victims-of-Homicides-and-Non-Fa/gumc-mgzr.

Data was extracted on December 7th, 2022. Map tiles by Stamen Design, under CC by 3.0. Data by © OpenStreetMap, under ODbL.

As discussed in the previous section on OLS linear regression, minimization of a loss function $J(c_0, c_1)$ is the basis for finding a line $y = c_0 + c_1 x$ which best fits a collection of data points $(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$. We show how minimization of a different loss function is basic to a particular method to cluster data into k groups called *k-means clustering*. We begin by describing three recursive methods for locating a minimum point of a loss function, the first of which is *gradient descent*.

10.3.2 Optimization by Gradient Descent

Recall that given a function $J(w_1, w_2, \ldots, w_s)$, the gradient of J (denoted ∇J) is the vector

$$\nabla J(w_1, w_2, \ldots, w_s) = \left\langle \frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2}, \ldots, \frac{\partial J}{\partial w_s} \right\rangle.$$

If $J(w_1, .w_2, \ldots, w_s)$ has a maximum or minimum at a differentiable interior point $\mathbf{w}^* = (w_1^*, w_2^*, \ldots, w_s^*)$, then

$$\nabla J(\mathbf{w}^*) = \mathbf{0}.$$

That is, all the partials of J must equal zero at $\mathbf{w}^*$. Hence, to find a max or min point $\mathbf{w}^*$ of J, we look for a solution to $\nabla J = \mathbf{0}$.

Let us consider the one-dimensional case to find a minimum point w^* of a function of one variable $J = J(w)$. Recall Newton's method, which says that to find a root w^* of $J(w)$, we can use the iterative algorithm

$$w_{n+1} = w_n - \frac{J(w_n)}{J'(w_n)},$$

where the sequence $w_n \to w^*$.

If we wish to find a root w^* of $J'(w)$ rather than of $J(w)$, Newton's method becomes

$$w_{n+1} = w_n - \frac{J'(w_n)}{J''(w_n)},$$

where the sequence $w_n \to w^*$. If σ is a positive constant near $\frac{1}{J''(w^*)} > 0$, then we can modify the last equation as

$$w_{n+1} = w_n - \sigma J'(w_n).$$

Generalizing to functions of more than one variable, we obtain the *gradient descent* recursive method to find a minimum point for J:

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \sigma \nabla J(\mathbf{w}_n),$$

where σ is a positive constant called the *learning rate.*

10.3.3 Example 3.1.

Let $J(w_1, w_2) = w_1^2 + 2w_1 + w_2^2 + 4w_2$, $\mathbf{w}_0 = (0, 0)$ and $\sigma = 0.1$. Find $\mathbf{w}_1$ and $\mathbf{w}_2$. To which point do we expect the gradient descent sequence $\mathbf{w}_n$ to converge?

Solution:
We have $\nabla J = (2w_1 + 2, 2w_2 + 4)$.

$$\mathbf{w}_1 \mathbf{w}_0 - \sigma \nabla J(\mathbf{w}_0) = (0, 0) - 0.1(2, 4) = (-0.2, -0.4)$$

$$\mathbf{w}_2 = \mathbf{w}_1 - \sigma \nabla J(\mathbf{w}_1) = (-0.2, -0.4) - 0.1(1.6, 3.2) = (-0.36, -0.72).$$

Note that $J = (w_1 + 1)^2 + (w_2 + 2)^2 - 5$ has a minimum point at $\mathbf{w}^* = (-1, -2)$. That point is where we would expect the gradient descent sequence $\mathbf{w}_n$ to converge.

We continue with minimizing a loss function $J(w_1, w_2, \dots, w_s)$ by introducing two additional methods of optimization.

Coordinate Descent

In contrast to gradient descent, *coordinate descent* seeks to minimize the value of J one coordinate at a time. For example, to minimize J, we begin with an initial point $\mathbf{w}_0$ which specifies the values for $(w_1, w_2, \dots, w_s)$ and look for the minimum value of J obtainable by varying w_1 and keep the remaining variables $w_2, \dots, w_s$ fixed. After finding the value of w_1 that gives a minimum J, we continue by varying the value of w_2 to further minimize J while keeping the newly obtained w_1 and the other variables fixed. We continue this process until the value of J remains unchanged (or only changes at each step by an amount less than a specified threshold) when cycling through all n variables.

Optimization by Coordinate and Block Descent

Example 3.2.

Use coordinate descent to minimize

$$J(w_1, w_2) = (w_1 - 1)^2 + (w_2 - 2)^2,$$

beginning with the initial point $\mathbf{w}_0 = (w_1, w_2) = (0, 0)$.

Solution:

First we fix $w_2 = 0$ and minimize $J = (w_1 - 1)^2 + 4$. The value $w_1 = 1$ gives the minimum J. Next, we fix $w_1 = 1$ and minimize $J = (w_2 - 2)^2$. The value $w_2 = 2$ gives the minimum. The result $J = 0$ is unchanged if we cycle through w_1 and then w_2. Hence, in this case, coordinate descent correctly gives $\mathbf{w} = (1, 2)$ as the point that minimizes J.

Block Descent

Block descent is similar to coordinate descent except we minimize J over blocks of variables rather than a single variable. The k-means clustering method is based on block descent, as we now explain.

K-MEANS CLUSTERING OPTIMIZATION PROBLEM

Find $\mathbf{z}_1, \dots, \mathbf{z}_k$ and $y_{i,j}$ that

$$minimize\, J(\mathbf{z}_1, \dots, \mathbf{z}_k; y_{i,j}) = \sum_{j=1}^{k} \sum_{i=1}^{m} y_{ij} \|\mathbf{x}_i - \mathbf{z}_j\|^2,$$

subject to the constraints $y_{i,j} \in \{0, 1\}$ for all i, j, and $\sum_{j=1}^{k} y_{i,j} = 1$ (each data point is assigned to exactly one centroid).

In this case, the loss function J is the total squared distance between data points and their assigned centers (prototypes).

Two blocks of variables are used in block descent minimization of J:

the *prototype variables* $\mathbf{z}_1, \dots, \mathbf{z}_k$ and

the *assignment variables* $y_{i,j}$ $(1 \le i \le m$ and $1 \le j \le k)$.

Block descent in this case results in a two-step iterative algorithm:

step 1) Fix the prototype variables, and then specify y_{ij} so that each data point $\mathbf{x}_i$ is assigned to the closest center $\mathbf{z}_j$.

step 2) Fix each assignment $y_{i,j}$ and choose the cluster center (prototype) to be the mean of the points in its cluster.

We continue this process until the value of J either is unchanged or only changes by some amount less than a specified threshold after completing the two steps in the algorithm.

Example 3.3.

Use k-means clustering with $k = 2$ to divide the points $\mathbf{x}_1 = (0, 0)$, $\mathbf{x}_2 = (10, 0)$, $\mathbf{x}_3 = (10, 1)$, and $\mathbf{x}_4 = (0, 1)$ into two clusters. Begin with the prototypes $z_1 = (0, 0)$ and $z_2 = (10, 0)$.

Solution:

i) Fix the prototypes $z_1 = (0, 0)$ and $z_2 = (10, 0)$; we have $y_{1,1} = 1$, $y_{1,2} = 0$, $y_{2,1} = 0$, $y_{2,2} = 1$, $y_{3,1} = 0$, $y_{3,2} = 1$, $y_{4,1} = 1$, and $y_{4,2} = 0$;

ii) Fix each assignment $y_{i,j}$; then the prototypes are $\mathbf{z}_1 = (0, 0.5)$ and $\mathbf{z}_2 = (10, 0.5)$.

iii)Fixing the prototypes $\mathbf{z}_1 = (0, .5)$, $\mathbf{z}_2 = (10, .5)$, the assignment variables are the same as before: $y_{1,1} = 1$, $y_{1,2} = 0$, $y_{2,1} = 0$, $y_{2,2} = 1$, $y_{3,1} = 0$, $y_{3,2} = 1$, $y_{4,1} = 1$, and $y_{4,2} = 0$.

iv) Fixing the assignment variables, the prototype variables are the same as before: $\mathbf{z}_1 = (0, 0.5)$ and $\mathbf{z}_2 = (10, 0.5)$.

Observing that $y_{1,1} = 1$ and $y_{4,1} = 1$, the points $\mathbf{x_1} = (0, 0)$ and $\mathbf{x_4} = (0, 1)$ are assigned to cluster 1. Furthermore, since $y_{2,2} = 1$ and $y_{3,2} = 1$, the points $\mathbf{x_2} = (10, 0)$ and $\mathbf{x_3} = (10, 1)$ are assigned to cluster 2.

Proof of Block Descent's Stepwise Minimization of J

Since J is the sum of squared distances between data points and their cluster centers, it is clear that if we fix the prototypes, J will be minimized by assigning each data point to the nearest prototype. We now show that for any fixed choice of assignment variables, the loss function J is minimized when each prototype is the mean of the points in its cluster.

Let $J = \sum_{j=1}^{k} O_j$ where $O_j = \sum_{i-1}^{m} y_{i,j} \|\mathbf{x}_i - \mathbf{z}_j\|^2$. Note that O_j can be computed using only the points in cluster j, since otherwise $y_{i,j} = 0$. Denote these points $\mathbf{u}_1, \dots, \mathbf{u}_{n_j}$. Then for each j, we minimize

$$O_j(z_{j,1}, \dots, z_{j,m}) = (u_{1,1} - z_{j,1})^2 + \cdots + (u_{1,m} - z_{j,m})^2 + (u_{2,1} - z_{j,1})^2 + \cdots + (u_{2,m} - z_{j,m})^2 + \cdots + (u_{n_j,1} - z_{j,1})^2 + \cdots + (u_{n_j,}$$

The condition $\nabla O_j(z_{j,1}, \dots, z_{j,m}) = (0, \dots, 0)$ will be satisfied if we choose $\mathbf{z}_j$ to be the average of the points $\mathbf{u}_1, \dots, \mathbf{u}_{n_j}$. For example,

$$\frac{\partial O_j}{\partial z_{j,1}} = -2(u_{1,1} - z_{j,1}) - 2(u_{2,1} - z_{j,1}) - \cdots - 2(u_{n_j,1} - z_{j,1}) = 0 \qquad \Rightarrow \sum_{t=1}^{n_j} u_{t,1} - n_j z_{j,1} = 0 \qquad \Rightarrow \qquad z_{j,1} = \frac{1}{n_j} \sum_{t=1}^{n_j} u_{t,1},$$

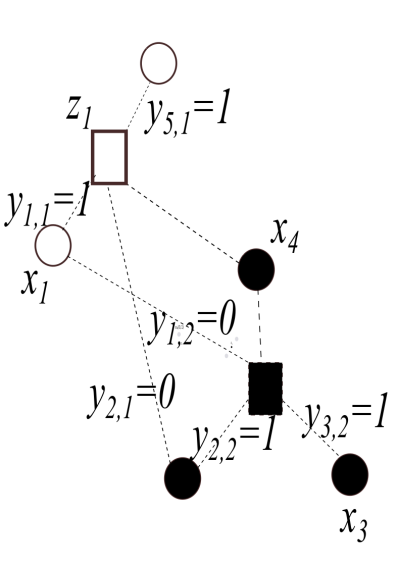
so the first coordinate of z_j (namely, $z_{j,1}$) is the average of the first coordinates of the points in its cluster. The same argument shows that each coordinate of $\mathbf{z}_j$ is the mean of the coordinates of points in its cluster.

10.3.4 Exercises

Exercises

1. Compute $\mathbf{w}_1$, $\mathbf{w}_2$, and $\mathbf{w}_3$ in the gradient descent sequence $\mathbf{w}_n$ obtained when minimizing $J(w_1, w_2) = w_1^2 + 2w_2^2$ with $\mathbf{w}_0 = (-5, -2)$ and learning rate $\sigma = 0.1$. To what point will the sequence $\mathbf{w}_n$ converge?

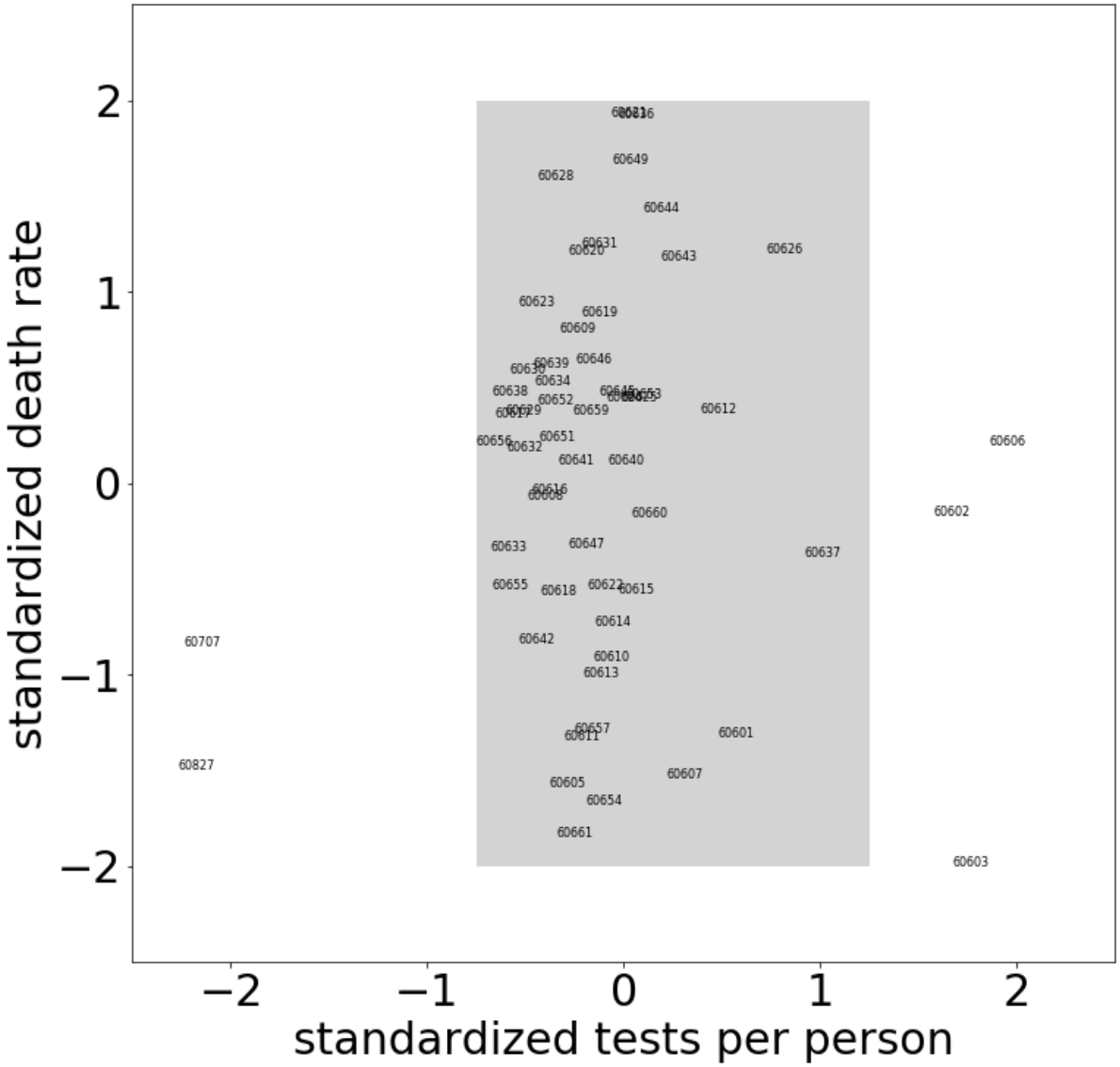
2. Use coordinate descent to minimize $J(w_1, w_2, w_3) = w_1^2 + (w_2 - 1)^2 + (w_3 - 2)^2$ beginning with the initial point $(w_1, w_2, w_3) = (0, 0, 0)$.
3. Use 2-means clustering to divide the points $\mathbf{x}_1 = (-1, -1)$, $\mathbf{x}_2 = (-1, 0)$, $\mathbf{x}_3 = (-1, 1)$, $\mathbf{x}_4 = (1, -1)$, $\mathbf{x}_5 = (1, 0)$, and $\mathbf{x}_6 = (1, 1)$ into two clusters. Begin with the prototypes $z_1 = (-1, 0)$ and $z_2 = (1, 0)$.
4. The graph in the figure below shows the positions of five data points $\mathbf{x}_1, \ldots, \mathbf{x}_5$, two cluster centers $\mathbf{z}_1$ and $\mathbf{z}_2$, as well as selected values for the assignment variables $y_{i,j}$. Supply the missing labels for the data points $\mathbf{x_2}$ and $\mathbf{x_5}$, cluster center $\mathbf{z_2}$, and assignment variables $y_{4,1}$ and $y_{4,2}$.



10.4 Dimension Reduction by Principal Component Analysis

Principal component analysis (PCA) is used to reduce high-dimensional data by projecting the data onto a much-lower-dimensional subspace. The subspace is chosen so that the variance of the projected data points is maximized.

For a simple example illustrating why maximizing the projected variance is useful, consider the scatterplot of COVID test rates and death rates by Chicago ZIP Code as shown in the figure below. For the ZIP Codes in the shaded portion of the scatterplot, we notice a much higher variance in the death rate than in the test rate. Hence, from the death rate for a ZIP Code, we would expect to learn more than from its test rate; the test rates are roughly the same for all ZIP Codes, but there is considerable spread in the death rates. Thus, if we wish to reduce the dimensionality of the data, it would be more informative to project onto the death rate rather than onto the test rate, since the former has greater variance. Dimension reduction is based on projections that maximize variance.



Chicago ZIP Codes shown in the shaded portion of the scatterplot have a variance of roughly 0.1 for the standardized test rate and 0.9 for the standardized death rate. (ZIP Code 60604, which includes the Chicago Board of Trade, Federal Building, and Chicago Symphony Center, has a standardized test rate of 5.7 and is not shown.)

10.4.1 Variance and Covariance for Mean-centered Data

Consider real data values **x** with mean $\bar{x}$ and **y** with mean $\bar{y}$.

i) The variance $Var(\mathbf{x})$ of **x** and variance $Var(\mathbf{y})$ of **y** are

$$Var(\mathbf{x}) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}, \qquad Var(\mathbf{y}) = \frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{n}.$$

ii) The covariance of **x** and **y** (denoted $Cov(\mathbf{x}, \mathbf{y})$ is

$$Cov(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n}.$$

Note that $Cov(\mathbf{x},\mathbf{y}) = Cov(\mathbf{y},\mathbf{x})$. Furthermore, for mean-centered data ($\bar{x} = 0$ and $\bar{y} = 0$), the formulas simplify to

$$Var(\mathbf{x}) = \frac{\sum_{i=1}^{n} x_i^2}{n} = \frac{1}{n}\,\mathbf{x}\cdot\mathbf{x}, \qquad Cov(\mathbf{x},\mathbf{y}) = \frac{\sum_{i=1}^{n} x_i y_i}{n} = \frac{1}{n}\,\mathbf{x}\cdot\mathbf{y}.$$

The variance measures how far the data are spread away from the mean, and the covariance measures the correspondence between x and y values.

10.4.2 Covariance Matrix

Consider an $n \times m$ data matrix $\mathbf{X}$ where each row represents a different data point (e.g., n people) and each column represents one of m different features (e.g., height, weight, age, etc.). We assume that the columns have all been mean-centered. Define the real *covariance matrix* $\mathbf{A}$ as

$$\mathbf{A} = \tfrac{1}{n}\,\mathbf{X}^T\mathbf{X}.$$

$\mathbf{A}$ must be symmetric since $\mathbf{A}^T = \frac{1}{n}\mathbf{X}^T(\mathbf{X}^T)^T = \frac{1}{n}\mathbf{X}^T\mathbf{X} = \mathbf{A}$. (Recall that $(\mathbf{MN})^T = \mathbf{N}^T\mathbf{M}^T$.)

As a consequence of the definition of $\mathbf{A}$ and as illustrated in the following example, the (i,i) entry of the covariance matrix $\mathbf{A}$ is the variance of the ith column of the data matrix $\mathbf{X}$, and the (i,j) entry ($i \neq j$) of the covariance matrix $\mathbf{A}$ is the covariance of the ith and j^{th} columns of the data matrix $\mathbf{X}$.

Example

Consider the data points $(-5,0)$, $(0,2)$, $(5,-2)$. The data are mean-centered. We have $Var(\mathbf{x}) = [(-5)^2+0^2+5^2]/3 = 50/3$, $Var(\mathbf{y}) = [0^2+2^2+(-2)^2]/3 = 8/3$, and $Cov(\mathbf{x},\mathbf{y}) = [(-5)(0)+0(2)+5(-2)]/3 = -10/3$.

For the corresponding data matrix

$$\mathbf{X} = \begin{pmatrix} -5 & 0 \\ 0 & 2 \\ 5 & -2 \end{pmatrix},$$

the mean of both columns is zero. The covariance matrix is

$$\begin{aligned}\mathbf{A} = \tfrac{1}{3}\mathbf{X}^T\mathbf{X} &= \frac{1}{3}\begin{pmatrix} -5 & 0 & 5 \\ 0 & 2 & -2 \end{pmatrix}\begin{pmatrix} -5 & 0 \\ 0 & 2 \\ 5 & -2 \end{pmatrix} \\ &= \frac{1}{3}\begin{pmatrix} 50 & -10 \\ -10 & 8 \end{pmatrix} = \begin{pmatrix} 50/3 & -10/3 \\ -10/3 & 8/3 \end{pmatrix}.\end{aligned}$$

As expected, the $(1,1)$ entry of $\mathbf{A}$ gives $Var(\mathbf{x}) = 50/3$, the $(2,2)$ entry gives $Var(\mathbf{y}) = 8/3$, and the $(1,2)$ and $(2,1)$ entries both give $Cov(\mathbf{x},\mathbf{y}) = Cov(\mathbf{y},\mathbf{x}) = -10/3$.

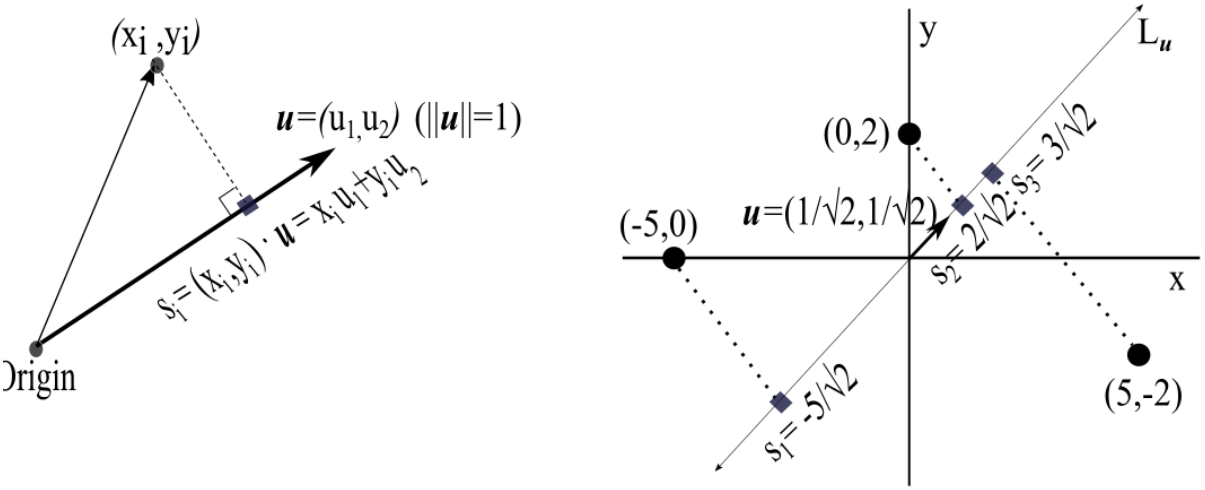
10.4.3 Projected Variance

Consider the data points $(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)$. We assume that the data are mean-centered, so $\bar{x} = \bar{y} = 0$. Our goal in this section is to find a simple matrix expression that can be used to compute the variance VAR of these data projected onto a line $L_{\mathbf{u}}$ through the origin determined by a unit vector $\mathbf{u} = (u_1,u_2)$. Referring to the left panel of the figure below, note that

$$\begin{aligned}\text{VAR} &= \frac{1}{n}\sum_{i=1}^{n} s_i^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i u_1 + y_i u_2)^2 \\ &= \frac{1}{n}\sum_{i=1}^{n} x_i^2 u_1^2 + 2x_i y_i u_1 u_2 + y_i^2 u_2^2 \\ &= u_1^2 Var(\mathbf{x}) + 2u_1 u_2 Cov(\mathbf{x},\mathbf{y}) + u_2^2 Var(\mathbf{y}).\end{aligned}$$

On the other hand, if we write $\mathbf{u}$ as a 2×1 column vector, VAR can also be computed as the matrix product $\mathbf{u}^T\mathbf{A}\mathbf{u}$:

$$\begin{aligned}\mathbf{u}^T\mathbf{A}\mathbf{u} &= (u_1\, u_2)\begin{pmatrix} Var(\mathbf{x}) & Cov(\mathbf{x},\mathbf{y}) \\ Cov(\mathbf{x},\mathbf{y}) & Var(\mathbf{y})\end{pmatrix}\begin{pmatrix} u_1 \\ u_2\end{pmatrix} \\ &= (u_1 Var(\mathbf{x}) + u_2 Cov(\mathbf{x},\mathbf{y}), \quad u_1 Cov(\mathbf{x},\mathbf{y}) + u_2 Var(\mathbf{y}))\begin{pmatrix} u_1 \\ u_2\end{pmatrix} \\ &= u_1^2 Var(\mathbf{x}) + 2u_1u_2 Cov(\mathbf{x},\mathbf{y}) + u_2^2 Var(\mathbf{y}) = \text{VAR}. \end{aligned} \tag{10.3}$$



(Left) Projected value s_i of a data point (x_i, y_i) onto a line through the origin determined by a unit vector $\mathbf{u} = (u_1, u_2)$. (Right) Projection of the 3 data points $(-5, 0)$, $(0, 2)$, and $(5, -2)$ onto the line $L_{\mathbf{u}}$ determined by the unit vector $\mathbf{u} = (1/\sqrt{2}, 1/\sqrt{2})$. The values s_1, s_2, and s_3 determine the variance $\text{VAR} = \frac{1}{3}(s_1^2 + s_2^2 + s_3^2)$ of the projected points on $L_{\mathbf{u}}$.

10.4.4 Example 4.2.

Let us project the three data points $(-5, 0)$, $(0, 2)$, and $(5, -2)$ onto the line $L_{\mathbf{u}}$ through the origin determined by the unit vector $\mathbf{u} = (u_1, u_2) = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ (see the right panel of the figure above). If we write $\mathbf{u}$ as a 2×1 column vector, the variance VAR of the projected points is equal to $\mathbf{u}^T\mathbf{A}\mathbf{u}$:

$$\begin{aligned}\text{VAR} &= (s_1^2 + s_2^2 + s_3^2)/3 = (25/2 + 4/2 + 9/2)/3 = 38/6 = 19/3, \\ \mathbf{u}^T\mathbf{A}\mathbf{u} &= (1/\sqrt{2} \quad 1/\sqrt{2})\begin{pmatrix} 50/3 & -10/3 \\ -10/3 & 8/3\end{pmatrix}\begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2}\end{pmatrix} \\ &= (40/3\sqrt{2} \quad -2/3\sqrt{2})\begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2}\end{pmatrix} = 40/6 - 2/6 = 38/6 = 19/3.\end{aligned}$$

10.4.5 Maximization of Projected Variance by Principal Component Analysis

Given an $n \times m$ data matrix X whose columns all have mean zero, the $m \times m$ covariance matrix $\mathbf{A}$ is the symmetric matrix

$$\mathbf{A} = \tfrac{1}{n}\mathbf{X}^T\mathbf{X}. \tag{10.4}$$

Since the $m \times m$ covariance matrix $\mathbf{A}$ is a real symmetric matrix, it has m linearly independent eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m \in \mathbf{R}^m$ with m real eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_m$. The first two principal components solve the following optimization problems:

(First principal component) Find vector $\mathbf{v}_1$ (written as an $m \times 1$ column vector) that will

$$maximize\, J(\mathbf{v}_1) = \mathbf{v}_1^T \mathbf{A} \mathbf{v}_1$$

such that $\|\mathbf{v}_1\|^2 = 1$. When the data is projected onto the line $\mathbf{L}_{\mathbf{v_1}}$ through the origin determined by the unit vector $\mathbf{v}_1$, $J(\mathbf{v}_1)$ is the variance of the data along that line.

The choice of $\mathbf{v}_1$ that maximizes the variance is a unit eigenvector of $\mathbf{A} = \frac{1}{n}\mathbf{X}^T\mathbf{X}$ that has the largest eigenvalue [Aggarwal 2020].

Second principal component Find vector $\mathbf{v}_2$ (written as an $m \times 1$ column vector) that will

$$maximize\, J(\mathbf{v}_2) = \mathbf{v}_2^T \mathbf{A} \mathbf{v}_2$$

such that $\|\mathbf{v}_2\|^2 = 1$ and $\mathbf{v_1} \cdot \mathbf{v}_2 = 0$. The vector $\mathbf{v}_1$ is the one obtained in the first step as the first principal component; $\mathbf{v}_2$ must be orthogonal to $\mathbf{v}_1$ and captures as much of the remaining variance in the data as possible that was not captured by $\mathbf{v}_1$.

More generally, to achieve our goal of dimensionality reduction we consider the following:

PRINCIPAL COMPONENT ANALYSIS OPTIMIZATION PROBLEM

The first k principal component vectors $\mathbf{v}_1, \ldots \mathbf{v}_k$ will

$$maximize\, J(\mathbf{v}_1, \ldots, \mathbf{v}_k) = \sum_{i=1}^{k} \mathbf{v}_i^T \mathbf{A} \mathbf{v}_i$$

subject to $\|\mathbf{v}_i\|^2 = 1$ for all $i = 1, \ldots, k$ and $\mathbf{v}_i \cdot \mathbf{v}_j = 0$ for all $i \neq j$. The vectors $\mathbf{v}_1, \ldots \mathbf{v}_k$ are orthonormal eigenvectors of $\mathbf{A}$ with the largest eigenvalues.

Typically, only a few of the non-negative eigenvalues of the positive semi-definite matrix $\mathbf{A}$ are relatively large. Assemble the k principal unit eigenvectors with the largest eigenvalues (principal components) as the columns of a $m \times k$ matrix $\mathbf{Q}_k$. The transformed data matrix $\mathbf{X}_k = \mathbf{X}\mathbf{Q}_k$ is an $n \times k$ matrix that is a reduced k-dimensional feature representation ($k << m$) of the original $n \times m$ data matrix $\mathbf{X}$. The PCA reduced matrix $\mathbf{X_k}$ captures as much of the variance of the original data matrix as is possible by a k-dimensional subspace [Aggarwal 2020].

Not all m features of the data set may be equally important in describing the data. PCA can be viewed as providing a sequence of best-fit subspaces of dimensions $k = 1, 2, \ldots, m-1$. These subspaces minimize the projected squared distances from the data and maximize the variance of the projected data. JNB Exercise 5 illustrates how PCA can be used to reduce high-dimensional ($N \times N$-pixel) vector representations of letter images. After projecting these N^2 dimensional vectors onto a two-dimensional subspace determined by the first two principal components, and then converting the projected vectors back to an image, we still maintain a good idea of the original letters. This would not be the case if we projected the image vectors onto a two-dimensional subspace determined by any two of the N^2 standard basis vectors $\mathbf{e_1} = (1, 0, 0, \ldots, 0)$, $\mathbf{e_2} = (0, 1, 0, \ldots, 0), \ldots, \mathbf{e_{N^2}} = (0, 0, 0, \ldots, 1)$.

10.4.6 Exercises

Exercises

1. Answer the following:

a) Use the covariance matrix to determine the variance and covariance of the x and y coordinates of the points $(-1, 3), (0, 0), (1, -3)$.

b) Use a principal component to determine on what line should we project the points $(-1, 3), (0, 0), (1, -3)$ to maximize the variance on that line.

2) Consider the $n \times m = 4 \times 3$ data matrix $\mathbf{X}$ defined as

$$\mathbf{X} = \begin{pmatrix} 2 & 2 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -1 \\ -2 & -2 & 0 \end{pmatrix}.$$

Each row represents a person, so there are data for $n = 4$ people. Each column represents a feature, so there are $m = 3$ features (e.g., height, weight, and age.) We assume that the columns are mean-centered and in standard units. Person 1 is tall and heavy, with average age. Person 2 has average height and weight and is older. Person 3 is average height and weight and is younger. Person 4 is relatively short and light, with average age.

a) Find the symmetric $m \times m$ covariance matrix $\mathbf{A} = \frac{1}{n}\mathbf{X}^T\mathbf{X}$

b) Find the eigenvalues and eigenvectors of $\mathbf{A}$ What are the first, second, and third principal components?

c) Explain how the variance is maximized by the principal components.

3) Consider the $n \times m = 4 \times 3$ data matrix $\mathbf{X}$ defined as

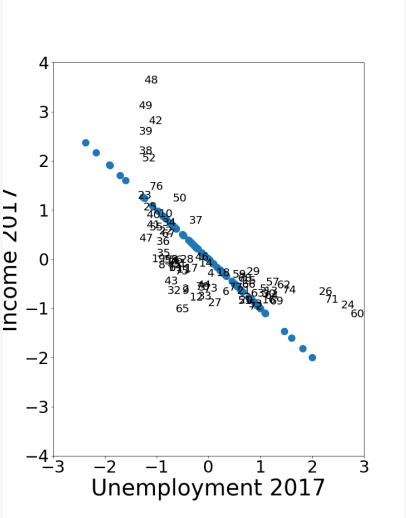
$$\mathbf{X} = \begin{pmatrix} 0 & 4 & 0 \\ 2 & 0 & 1 \\ -2 & 0 & -1 \\ 0 & -4 & 0 \end{pmatrix}.$$

a) Without calculating eigenvectors, conjecture the first two principal components.

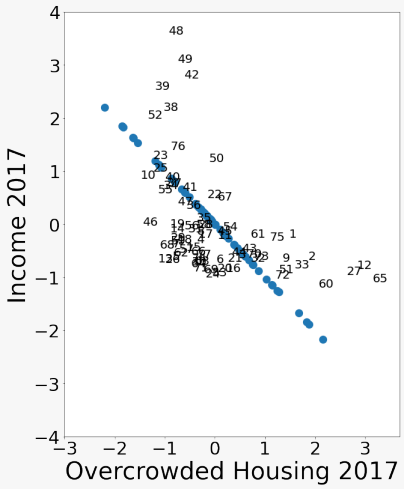
b) Check your answer to a) by computing the eigenvectors of the covariance matrix $\mathbf{A} = \frac{1}{n}\mathbf{X}^T\mathbf{X}$.

4. Answer the following:

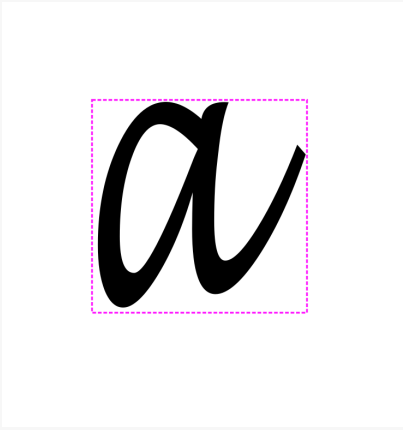
a) Using the JNB PRINCIPAL COMPONENT ANALYSIS at https://tinyurl.com/2neb4z4c, the figure below shows a scatterplot of standardized hardship index data for unemployment over age 16 and per capita income [UIC Great Cities 2017]. The original data is labeled by the number of the Chicago community area (1–77). What do the projected points indicate?



b) The figure below shows overcrowded housing and per capita income. Find a difference between this graph and the graph for unemployment in part a)

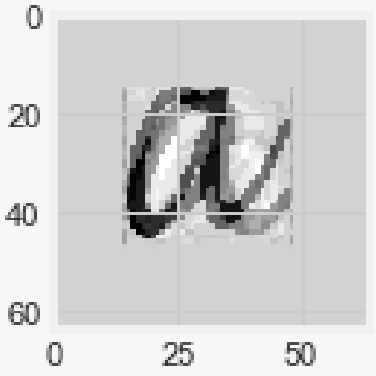


4.5 a) We can illustrate the dimensionality reduction capacity of PCA by applying it to character recognition. The Jupyter Notebook PRINCIPAL COMPONENT ANALYSIS at https://tinyurl.com/2neb4z4c shows how to take a collection of eight 64×64 bit images of the letters `a'` `and` b', and project each letter onto a two-dimensional subspace determined by the first two principal components. Referring to the figures below, explain why the original image is $64^2 = 4096$-dimensional and the PCA image is only two-dimensional.



Original 64x64 pixel image.

2-dimensional representation is obtained by projecting the original image onto a 2-dimensional subspace determined

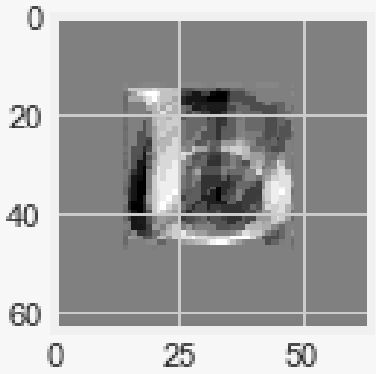


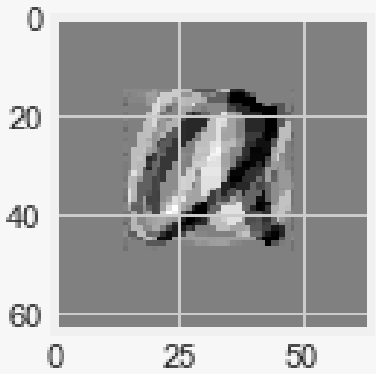
by the first two principal components.

b) The basis vectors $\mathbf{v}_1$, $\mathbf{v}_2$ for the two-dimensional subspace determined by PCA correspond to the images in the figures below. The reduced dimensional representation of the letter 'a' in part **a**) is the linear combination $2560.3\mathbf{v}_1 - 2207.85\mathbf{v_2}$. What would happen if we used two standard basis vectors $\mathbf{e}_i$, $\mathbf{e}_j$ to determine the subspace of projection?

For example, what would happen if we choose $\mathbf{e_1} = (1, 0, 0, \ldots, 0)$ and $\mathbf{e_2} = (0, 1, 0, 0, \ldots, 0)$?

Images corresponding to the PCA 2D basis vectors $\mathbf{v}_1, \mathbf{v}_2$.

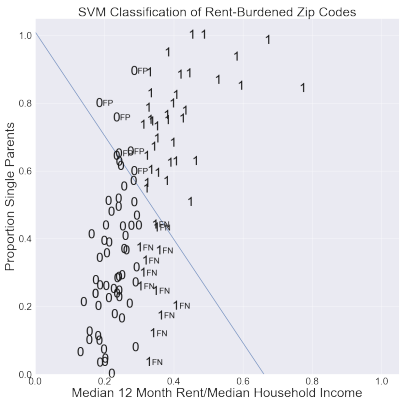




10.5 Binary Classification of Labeled Data by Support Vector Machines

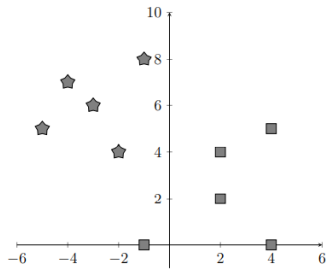
In this section, we consider data points that are labeled as belonging to one of two classes. Our goal is to determine a line (more generally, a hyperplane) that correctly separates the data. That is, all the points in one class are on one side of the line (hyperplane), and all the points in the other class are on the other side. For example, the scatterplot in the figure below is an example of linearly separable data, meaning that the data can be correctly separated by a line. The scatterplot shows for each Chicago ZIP Code the median proportion of income spent on rent vs. % of single-parent households. The labels indicate 1 = rent-burdened (median 12-month rent is more than 30% of median annual income) or 0 = not rent-burdened. As such, the line $x = 0.3$ separates the data. Points on one side of the line are in one class and points on the other in the other class.

As shown in the figure below, a line that does not correctly separate the labeled data will create false positives (FP) and/or false negatives (FN). An FP is a point that the line indicates to be in class 1 but which in fact is in class 0; an FN is a point that the line indicates to be in class 0 but which is actually in class 1.

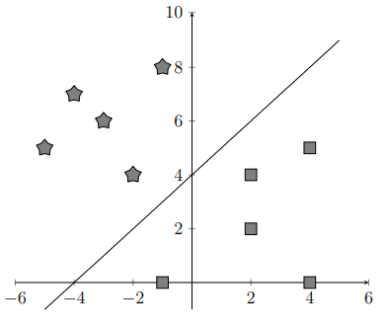

SVM Classification of Rent-Burdened Zip Codes
Proportion Single Parents
Median 12 Month Rent/Median Household Income

10.5.1 Intuition Underlying SVM

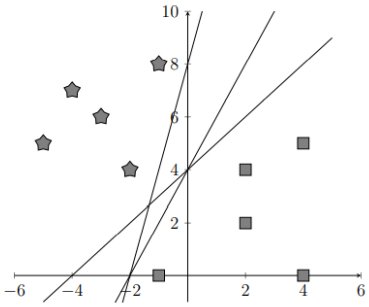
For simplicity, let us consider a small number of data points with each point labeled as belonging to one of two distinct groups indicated by a pentagon or square. Our goal is to find a boundary line that separates the two groups.



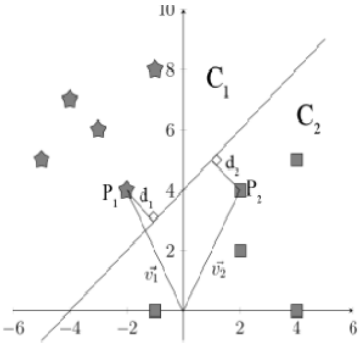
We might separate the data with a line as shown below.



Likewise, if we had data in a 3D coordinate system, we could separate the data by a 2D plane. Such a boundary in any dimensional space that separates the labeled points into 2 groups is called a hyperplane. As shown below, multiple hyperplanes can correctly separate data points into 2 groups.

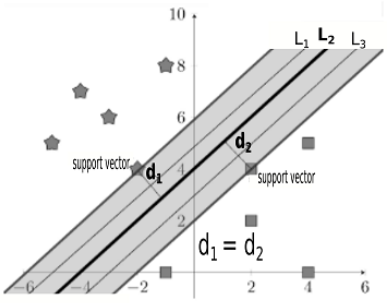


It's clear that some lines (hyperplanes) separate the data better than others. However, the mathematical description of the optimal line is not necessarily obvious. To describe this optimal hyperplane, we need to introduce two new terms, **support vectors** and **margins**. A support vector is a vector whose terminal point P_n in group C_n $(n = 1, 2)$ is closest to the hyperplane. Since we have two groups, we'll have at least two support vectors. The margin of a hyperplane is the sum $d_1 + d_2$ of the distances between the two closest points and the hyperplane.



In the figure above, P_1 and P_2 are support vectors.

The figure below shows that while the parallel separating hyperplanes L_1, L_2, and L_3 all have the same margin, only the SVM hyperplane L_2 is equidistant $(d_1 = d_2)$ from support vectors in both classes.



An ideal hyperplane would separate the data most clearly, and thus we would not want the hyperplane to be very close to any of the data points. Instead, the ideal hyperplane is as far from both classes of data as possible while continuing to separate the two groups. Thus, we should aim to maximize the margin around the hyperplane. However, given a fixed margin, multiple hyperplanes correctly separate the data, as shown in the figure above. The separating line L_1 is close to the star data but well-separated from the square data. L_1 is not an ideal hyperplane because a small error in the star data could result in incorrectly separated points. In order to prevent small variations in the data from affecting the grouping of points, we should look for a hyperplane that is well-separated from both classes, namely, the hyperplane that has a maximized margin and is equidistant from support vectors in the two classes.

10.5.2 Mathematical Formalism

Consider the binary classification of a labeled training data set consisting of n data points (observations). Each data point has m coordinates (features) and is represented by a row vector in $\mathbf{R}^m$. Binary class membership is indicated by the labels $y_i \in \{+1, -1\}$. The classification problem that we consider involves

i) finding a hyperplane that separates points by their class, using a labeled training set (see table below) to supervise the classification; and ii) using the hyperplane to make a prediction as to the class of a new point given just its features.

Observation	Feature 1	...	Feature m	Class Label
$\mathbf{x}_1$	$x_{1,1}$	...	$x_{1,m}$	y_1
$\vdots$				$\vdots$
$\mathbf{x}_n$	$x_{n,1}$	...	$x_{n,m}$	y_n

10.5.3 Separating Hyperplanes

The goal of linear separation is to find components of a vector $\mathbf{w} = \langle w_1, \dots, w_m \rangle$ and real constant b such that the hyperplane $\pi(\mathbf{w}, b)$ defined by the equation $\mathbf{w} \cdot \mathbf{x} + b = w_1 x_1 + w_2 x_2 + \cdots + w_m x_m + b = 0$ is a linear separator of the labeled training data $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$. That is, for all $i = 1, \dots, n$:

$\mathbf{w} \cdot \mathbf{x}_i + b > 0$ if $y_i = +1$, and

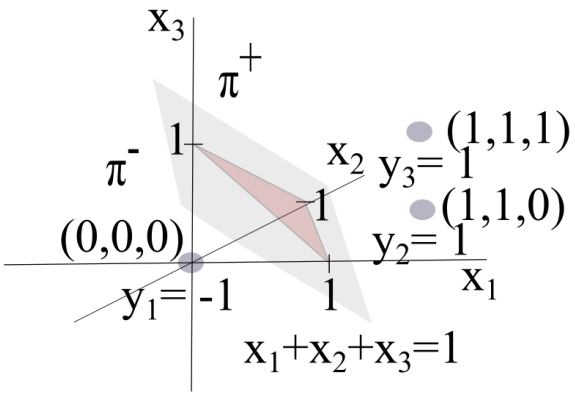
$\mathbf{w} \cdot \mathbf{x}_i + b < 0$ if $y_i = -1$.

For the hyperplane $\pi(\mathbf{w}, b)$, $\mathbf{w}$ is called the *normal vector* and b the *intercept*.

Define the half-spaces

$$\pi^+(\mathbf{w}, b) = \{\mathbf{x} \mid \mathbf{w} \cdot \mathbf{x} + b > 0\}, \qquad \pi^-(\mathbf{w}, b) = \{\mathbf{x} \mid \mathbf{w} \cdot \mathbf{x} + b < 0\}.$$

The hyperplane $\pi(\mathbf{x}, b)$ separates the data $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m$ if for each $i = 1, \dots, m$, $\mathbf{x}_i \in \pi^+$ if $y_i = 1$ and $\mathbf{x}_i \in \pi^-$ if $y_i = -1$.



In the figure above, the hyperplane $\pi((1,1,1),-1)$ with normal vector $\mathbf{w} = (1,1,1)$ and intercept $b = -1$ is the set of all points in (x_1, x_2, x_3) which satisfies the equation $x_1 + x_2 + x_3 - 1 = 0$. The half-spaces are

$$\pi^+ : x_1 + x_2 + x_3 - 1 > 0$$
$$\pi^- : x_1 + x_2 + x_3 - 1 < 0$$

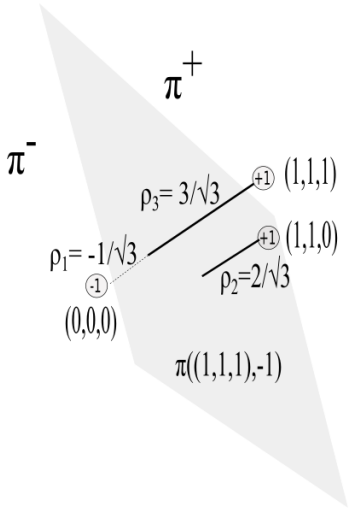
The data point $\mathbf{x}_1 = (0,0,0)$ with label $y = -1$ is in π^-, and the data points $\mathbf{x}_2 = (1,1,0)$ and $\mathbf{x}_3 = (1,1,1)$ with label $y = 1$ are in π^+. The hyperplane $\pi((1,1,1),-1)$ correctly separates the three labeled training points.

10.5.4 Signed Distance

The signed distance, denoted $\rho(\mathbf{x}, \pi)$, from a data point (vector) $\mathbf{x}$ to the hyperplane $\pi(\mathbf{w}, b)$ is

$$\rho(\mathbf{x}, \pi) = \frac{\mathbf{w} \cdot \mathbf{x} + b}{\|\mathbf{w}\|}.$$

Signed-distance is such that $\rho(\mathbf{x}, \pi) > 0$ when $\mathbf{x} \in \pi^+$ and $\rho(\mathbf{x}, \pi) < 0$ when $\mathbf{x} \in \pi^-$. These signed distances are shown in the figure below.



Let C_1 be the subset of data points with label $y = 1$ and C_2 be the subset of data points with label $y = -1$. The (non-negative) distance $\rho(\pi, C)$ between hyperplane π and class C is defined as

$$\rho(C, \pi) = \min_{\mathbf{x} \in C} \mid \rho(\mathbf{x}, \pi) \mid .$$

The *margin* $m(\pi, C_1, C_2)$ of a hyperplane π that separates classes C_1 and C_2 is defined as the sum of the (non-negative) distances between π and classes C_1 and C_2:

$$m(C_1, C_2, \pi) = \rho(C_1, \pi) + \rho(C_2, \pi).$$

In our example, $C_1 = \{(1,1,0),(1,1,1)\}$, $C_2 = \{(0,0,0)\}$, $\rho(C_1, \pi) = 2/\sqrt{3}$, and $\rho(C_2, \pi) = 1/\sqrt{3}$. The margin is therefore $m(C_1, C_2, \pi) = 1/\sqrt{3} + 1/\sqrt{3} = 2/\sqrt{3}$.

10.5.5 Optimization for Linearly-Separable Data

For classifying data points, SVM seeks a hyperplane $\pi(\mathbf{w}, b)$ that

i) correctly separates the training data,

ii) has maximum margin (this is desirable to allow for measurement error or noise in the data), and

iii) is equidistant from both classes of training data.

The support vectors are the data points in each class that are closest to the SVM hyperplane. Under the assumption that the training data are linearly separable, our goal to separate the data using a hyperplane $\pi(\mathbf{w}, b)$ is achieved by the following:

SVM OPTIMIZATION PROBLEM (separable training data)

Find $\mathbf{w}$ and b that will

$$minimize\, J = \frac{1}{2}\|\mathbf{w}\|^2$$

subject to the constraints

i) , $\mathbf{w} \cdot \mathbf{x} + b \geq 1$ for all $\mathbf{x} \in C_1$

ii) , $\mathbf{w} \cdot \mathbf{x} + b \leq -1$ for all $\mathbf{x} \in C_2$

where C_1 and C_2 are (non-empty) classes of labeled training vectors.

A result from convexity theory says that there will be a unique solution if the training classes C_1, C_2 are linearly separable, and no solution if they are not linearly separable [Aggarwal 2020]. A vector $\mathbf{x}^*$ is a support vector if either $\mathbf{w} \cdot \mathbf{x}^* + b = 1$ or $\mathbf{w} \cdot \mathbf{x}^* + b = -1$.

Note that by minimizing $J = \frac{1}{2}\|\mathbf{w}\|^2$, we are maximizing $d = \frac{1}{\|\mathbf{w}\|}$. From constraint **(i)**, we have $\rho(\mathbf{x}, \pi) = \frac{\mathbf{w}\cdot\mathbf{x}+b}{\|\mathbf{w}\|} \geq \frac{1}{\|\mathbf{w}\|} = d$ for all $\mathbf{x} \in C_1$. From constraint **(ii)**, we have $\rho(\mathbf{x}, \pi) = \frac{\mathbf{w}\cdot\mathbf{x}+b}{\|\mathbf{w}\|} \leq \frac{-1}{\|\mathbf{w}\|} = -d$ for all $\mathbf{x} \in C_2$. The margin is therefore $2d$. Hence, the minimization of $J = \frac{1}{2}\|\mathbf{w}\|^2$ maximizes the margin $2d = \frac{2}{\|w\|}$. For our example, using the Jupiter Notebook at https://tinyurl.com/ypasckyh, the support vectors are $\langle 0,0,0\rangle$ and $\langle 1,1,0\rangle$. This gives us the SVM separating hyperplane $\pi((1,1,0), -1)$ or $x_1 + x_2 - 1 = 0$ with a margin of $\sqrt{2}$. This margin is indeed larger than the margin $2/\sqrt{3}$ that we obtained with our previous separating hyperplane $x_1 + x_2 + x_3 - 1 = 0$.

10.5.6 Optimization for Non-Separable Data

In general, the training data might not be linearly separable. In this case, our goal to find a hyperplane $\pi(\mathbf{w}, b)$ that classifies the labeled data is based on the following:

SVM OPTIMIZATION PROBLEM (non-separable training data)

Find $\mathbf{w}$ and b that will

$$minimize\, J(\mathbf{w}.b) = \frac{1}{2}\sum_{i=1}^{n} \max\{0, 1 - y_i(\mathbf{w} \cdot \mathbf{x}_i + b)\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

where $\lambda > 0$ is a constant called the regularization parameter, and J is called the SVM L_2 loss function.

For a given choice of **w** and b, a prediction $\hat{y}_i = \text{sign}(\mathbf{w} \cdot \mathbf{x}_i + b)$ is correct if

i) $\mathbf{w} \cdot \mathbf{x}_i + b > 0$ if $y_i = +1$,

ii) $\mathbf{w} \cdot \mathbf{x}_i + b < 0$ if $y_i = -1$.

The summation term included in this L_2 SVM loss function $J(w_1, \ldots, w_n, b)$ is such that:

-a correct prediction contributes at most 1/2 to the value of J;

-an incorrect prediction contributes at least 1/2 to J; and

-correctly predicted points which are 'well-separated' from the separating hyperplane (i.e. $1 - y_i(\mathbf{w} \cdot \mathbf{x}_i - k) \leq 0$) do not contribute anything to J.

The term $\frac{\lambda}{2}\|\mathbf{w}\|^2$ relates to our previous discussion of maximizing the margin. The regularization parameter λ adjusts whether widening the margin or making correct classifications should be weighted more heavily.

Exercise 3) uses a JNB to illustrate how SVMs can be used in the binary classification of facial images. Using a training set consisting of $32 \times 32 = 1024$-dimensional vectors representing 32×32 pixel images of former Chicago Mayors Harold Washington and Richard M. Daley, an SVM hyperplane is used to predict which mayor is depicted in a set of test images of the two mayors.

10.5.7 Exercises

Exercises

1) Let $\mathbf{w} = (1, 2) \in \mathbf{R}^2$ and $b = -1$.

a) What is the hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$?

b) What are the half spaces $(\mathbf{R}^b_\pi)^+$ and $(\mathbf{R}^b_\pi)^-$?

2) Answer the following:

a) Show that $\pi(\mathbf{w}, b) = \pi(k\mathbf{w}, kb)$ for any $k \neq 0$

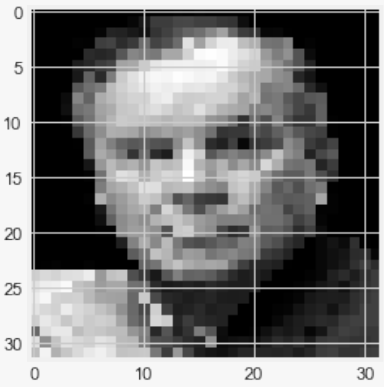
b) Show that the signed distance from the origin $\mathbf{x} = \mathbf{0}$ to the hyperplane $\pi(\mathbf{w}, b)$ is equal to $\frac{b}{\|\mathbf{w}\|}$.

c) Explain why there are an infinite number of separating hyperplanes with the same margin as the SVM hyperplane. How is the SVM hyperplane determined?

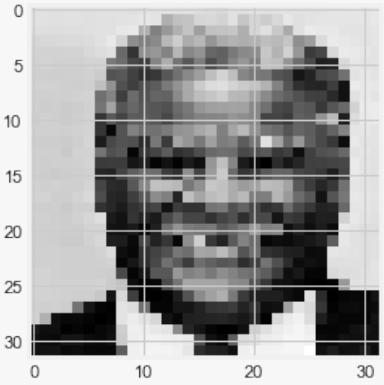
d) In the case where all points in the training set are correctly classified and well-separated, how does the SVM L_2 loss function simplify, and what results from it being minimized?

3) Binary SVM classification can be used for facial recognition, as demonstrated in the Jupyter Notebook SUPPORT VECTOR MACHINES (at https://tinyurl.com/2neb4z4c). An image can be represented as an $N \times N$ matrix **X** where the i, j entry is a number representing the grayscale value of the pixel in row i and column j. From this matrix, we can form an $N^2 \times 1$ column vector by concatenating rows in the matrix. Suppose that we have a set of images of two Chicago mayors, Mayor Daley and Mayor Washington. Each image is represented as a vector in $\mathbf{R}^{N^2}$ and is labeled 0 for Mayor Daley and 1 for Mayor Washington. An SVM hyperplane separates the labeled binary training data. Given a new image of either Mayor Washington or Mayor Daley (such as shown in the figures below, the SVM hyperplane can be used to predict whether the image is in class 0 = Daley or class 1 = Washington.

32×32 pixel images of former Chicago mayor Daley



and Washington



Use the JNB to make an SVM classification of images of Michelle Obama and Hillary Clinton (both first ladies were raised in Chicago).

10.5.8 Conclusion

In this chapter, we have seen how linear algebra and optimization provide a framework for four types of machine-learning data analysis: ordinary least squares (OLS) linear regression, k-means clustering of unlabeled data, dimensionality analysis by principal component analysis, and binary classification of labeled data by support vector machines (SVM). Aggarwal [2020] provides an extensive in-depth treatment of this subject, and Williams [2018] is a good introduction to additional ways linear algebra is useful in data analysis.

Data analysis is becoming increasingly automated with the development of machine learning tools, including Python JNBs [VanderPlas 2017]. In addition to being adept in the use of such computational tools, a key to proficiency as a data scientist is to have an inquisitive love for data as well as a good understanding of the mathematics underlying a given method of analysis.

Our team also holds the view that data analysis can and should support efforts to solve pressing societal issues being tackled by organizations working in disadvantaged communities. The housing data which we used to introduce SVM, is an instructive example of applying data analysis tools to complex problems involving equitable distribution of scarce resources. In pondering why SVM did not give the vertical line which correctly separates rent-burdened from non-rent-Chicago ZIP Codes, we realized that the L_2 SVM loss function penalizes data points that are correctly classified but are too close to the line (overfitting). In this case, if the default *regularization parameter* $C = 1$ in Python's Support Vector Classifier SVC() were increased to $C = 1000$, we would obtain the separating vertical hyperplane. (C is inversely proportion to λ in the SVM loss function in **Section 5.7**). Using the default SVM classifier with $C = 1$ results in misclassifications: Accuracy = 0.7 (42 out of 60 total predictions were correct), Recall = 0.75 (36 of 48 rent-burdened

ZIP Codes were correctly predicted), Precision = 0.86 (of the 42 ZIP Codes predicted to be rent-burdened, 36 were correctly predicted), and F-score = 0.8 (geometric mean of accuracy and precision). If the default model were the basis for a decision on whether to grant rent support, the false negatives (FN = 12) would deny support from qualified ZIP Codes, and false positives (FP = 6) would result in an inequitable use of limited resources. Understanding the mathematics underlying the algorithm can help identify potential dangers in accepting the default ML output at face value.

Knowledge of the real-world context under consideration is paramount. Although the vertical line $x = 0.3$ correctly separates the data and thus has 100% accuracy, recall, and precision, there are a dozen non-rent-burdened ZIP Codes close to the cutoff value of 0.3. Gentrification can result in rapid rental increases that negatively impact low-income areas by creating additional rent-burdened households. Yet, since incomes may also increase in gentrifying areas, ZIP Codes close to but below the threshold line might not cross over into the rent-burdened category. Similarly, ZIP Codes above the threshold might move below in spite of the additional rent burden on low-income households. Those most in need of housing support might be overlooked if funding decisions are based on a superficial machine-learning analysis of aggregated data. A merely mathematical classification of such data might hide or contribute to inequity if not accompanied by a close working knowledge of the actual real-life context.

10.5.9 References

Aggarwal, C. 2020. *Linear Algebra and Optimization for Machine Learning: A Textbook*. Cham, Switzerland: Springer Nature.

Amdat, W.C. 2021. The Chicago Hardship Index: An Introduction to Urban Inequity. Journal of Statistics and Data Science Education. https://www.researchgate.net/publication/361130606_The_Chicago_Hardship_Index_An_Introduction_to_Urban_Inequity.

Epperly, E. 2022. Don't solve the normal equations. https://www.ethanepperly.com/index.php/2022/07/26/dont-solve-the-normal-equations/

Great Cities Institute. 2017. Hardship index fact sheet.https://greatcities.uic.edu/wp-content/uploads/2019/12/Hardship-Index-Fact-Sheet-2017-ACS-Final-1.pdf.

Sun, S., Cao, Z., Zhu, H., and Zhao J. 2019. A Survey of Optimization Methods from a Machine Learning Perspective. https://arxiv.org/pdf/1906.06821.pdf#:~:text=Building models and constructing reasonable,to solve the optimization problem.

Toth, Geilert. 2020. Scaling, centering, and standardization. https://www.datasklr.com/ols-least-squares-regression/scaling-centering-and-standardization.

VanderPlas, J. 2017. *Python Data Science Handbook: Essential Tools for Working with Data*. Sebastopol, CA O'Reilly.

Williams, G. 2018. *Linear Algebra with Applications (9e).} Burlington, MA: Jones & Bartlett.*

Acknowledgements

The project team is grateful to UMAP Journal Editor Paul Campbell for helpful guidance as to the design and content of this chapter, and also to anonymous referees for suggesting improvements to the original manuscript.

10.6 Solution to Exercises

10.6.1 OLS Linear Regression

Problem 1) The system is

$$\begin{aligned} a + b &= 0, \\ a + 4b &= 5, \\ a + 7b &= 8; \end{aligned}$$

or, in matrix form,

$$\begin{pmatrix} 1 & 1 \\ 1 & 4 \\ 1 & 7 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 0 \\ 5 \\ 8 \end{pmatrix}.$$

The normal equations are

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 4 & 7 \end{pmatrix} \begin{pmatrix} 0 \\ 5 \\ 8 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 4 & 7 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 4 \\ 1 & 7 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix},$$

or

$$\begin{pmatrix} 13 \\ 76 \end{pmatrix} = \begin{pmatrix} 3 & 12 \\ 12 & 66 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}.$$

Solving the normal equations gives the least-squares solution $a = -1$, $b = 4/3$.

Problem 2)

The system is

$$\begin{aligned} a - b + c &= 1, \\ a &= 0, \\ a + b + c &= 2, \\ a + 2b + 4c &= 3; \end{aligned}$$

or, in matrix form,

$$\begin{pmatrix} 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 2 \\ 3 \end{pmatrix}.$$

The normal equations are

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \\ 1 & 0 & 1 & 4 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \\ 1 & 0 & 1 & 4 \end{pmatrix} \begin{pmatrix} 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix},$$

or

$$\begin{pmatrix} 6 \\ 7 \\ 15 \end{pmatrix} = \begin{pmatrix} 4 & 2 & 6 \\ 2 & 6 & 8 \\ 6 & 8 & 18 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix}.$$

Solving the normal equations gives the least-squares solution $a = 3/5, b = 3/10$, and $c = 1/2$.

10.6.2 K-means clustering

Problem 1) $\mathbf{w}_1 = (-4, -1.2)$, $\mathbf{w}_2 = (-3.2, -0.72)$ and $\mathbf{w}_3 = (-2.56, -0.432)$. The gradient descent sequence will converge to the minimum point at $(0, 0)$. To prove this, note that if we let $\mathbf{w}_n = (x_n, y_n)$, then $\mathbf{w}_n = \mathbf{w}_{n-1} - 0.1\nabla\mathbf{w}_{n-1}$ implies that $x_n = 0.8x_{n-1}$ and $y_n = 0.6y_{n-1}$. It follows that $\lim_{n\to\infty} x_n = \lim_{n\to\infty} y_n = 0$.

Problem 2)

i)Fix the prototypes $z_1 = (-1, 0)$ and $z_2 = (1, 0)$. Then we have $y_{1,1} = 1$, $y_{1,2} = 0$, $y_{2,1} = 1$, $y_{2,2} = 0$, $y_{3,1} = 1$, $y_{3,2} = 0$, $y_{4,1} = 0$, $y_{4,2} = 1$, $y_{5,1} = 0$, $y_{5,2} = 1$, $y_{6,1} = 0$, $y_{6,2} = 1$.

ii) Fix each of these assignments, then the prototypes are unchanged: $\mathbf{z}_1 = (-1, 0)$, $\mathbf{z}_2 = (1, 0)$.

iii) Fix the prototypes and the assignments are as before.

Thus, the points $(-1, -1)$, $(-1, 0)$, $(-1, 1)$ are assigned to cluster 1, and the points $(1, -1)$, $(1, 0)$, and $(1, 1)$ to cluster 2.

Problem 3)

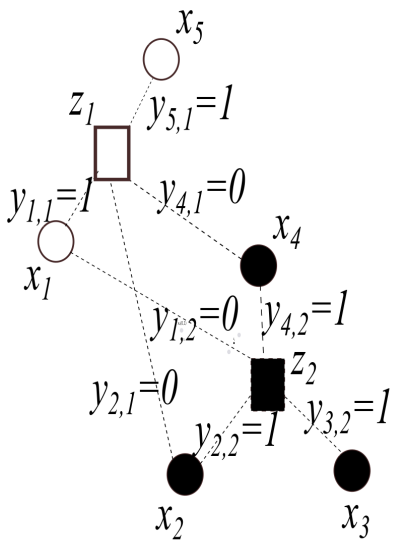
i) Fix the prototypes $z_1 = (-1, 0)$ and $z_2 = (1, 0)$. Then we have $y_{1,1} = 1$, $y_{1,2} = 0$, $y_{2,1} = 1$, $y_{2,2} = 0$, $y_{3,1} = 1$, $y_{3,2} = 0$, $y_{4,1} = 0$, $y_{4,2} = 1$, $y_{5,1} = 0$, $y_{5,2} = 1$, $y_{6,1} = 0$, $y_{6,2} = 1$.

ii) Fix each of these assignments, then the prototypes are unchanged: $\mathbf{z}_1 = (-1, 0)$, $\mathbf{z}_2 = (1, 0)$.

iii) Fix the prototypes and the assignments are as before.

Thus, the points $(-1, -1)$, $(-1, 0)$, $(-1, 1)$ are assigned to cluster 1, and the points $(1, -1)$, $(1, 0)$, and $(1, 1)$ to cluster 2.

Problem 4)



10.6.3 Dimension Reduction by PCA

1)

a)

The data matrix is

$$\mathbf{X} = \begin{pmatrix} -1 & 3 \\ 0 & 0 \\ 1 & -3 \end{pmatrix}.$$

The covariance matrix is

$$\mathbf{V} = \frac{1}{3}\begin{pmatrix} 2 & -6 \\ -6 & 18 \end{pmatrix} = \begin{pmatrix} 2/3 & -2 \\ -2 & 6 \end{pmatrix}$$

Thus, $Var(\mathbf{x}) = 2/3$, $Var(\mathbf{y}) = 6$, and $Cov(\mathbf{x}, \mathbf{y}) = -2$.

b) The eigenvalues are 0 and 10. The principal component is determined by the largest eigenvalue $\lambda = 10$, which has eigenvector $(1, -3)$. This vector determines the line $y = -3x$, which is the desired line.

2)

a)

$$\mathbf{A} = \begin{pmatrix} 2 & 2 & 0 \\ 2 & 2 & 0 \\ 0 & 0 & 1/2 \end{pmatrix}.$$

b)

The eigenvalues of $\mathbf{A}$ are $\lambda_1 = 4$, $\lambda_2 = 1/2$, $\lambda_3 = 0$, with corresponding unit eigenvectors for $\mathbf{v}_1 = (1/\sqrt{2}, 1/\sqrt{2}, 0)^T$, $\mathbf{v}_2 = (0, 0, 1)^T$, and $\mathbf{v}_3 = (1/\sqrt{2}, -1/\sqrt{2}, 0)^T$

The first principal component of $\mathbf{X}$ is given by $\lambda_1, \mathbf{v}_1$, the second principal component of is given by $\lambda_2, \mathbf{v}_2$, and the third principal component is given by $\lambda_3, \mathbf{v}_3$.

c)

The variance of the data along the line through the origin determined by $\mathbf{v}_1$ is the maximum among all lines through the origin.

The total variance of the data (sum of squared distances to the origin) is the sum of the variances by projecting the data onto the three lines through the origin determined by the mutually orthogonal unit vectors $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$.

The variances along each of these lines decrease. Each captures more of the variance in an optimal way. The first principal component captures the most amount of variance projecting the data onto a line through the origin (1D subspace). This line is determined by $\mathbf{v}_1$. The first two principal components capture the most variance if the data is projected onto a plane through the origin (2D subspace): the plane is determined by the vectors $\mathbf{v}_1$ and $\mathbf{v}_2$.

3)

a) By inspection, the greatest variance occurs in the y-coordinate, so we expect the first principal component to be in the direction of the line determined by $(0, 1, 0)$. In the orthogonal direction (i.e., the xz-plane), we see that the data lie on the line $x = 2z$. Thus, we expect the second principal component to be in the direction of the line determined by $(2, 0, 1)/\sqrt{5}$.

b)

$$\mathbf{A} = \begin{pmatrix} 2 & 0 & 1 \\ 0 & 8 & 0 \\ 1 & 0 & 1/2 \end{pmatrix}.$$

The first principal component is given by $\lambda_1 = 8$ with eigenvector $(0, 1, 0)$.

The second principal component is given by eigenvector $\lambda_2 = 5/2$ with eigenvector $(2, 0, 1)/\sqrt{5}$.

4)

a) The line along which the projected points have the greatest variance.

b) There is a difference in the outliers–of significance in considering the most vulnerable.

5)

a) The original image is represented by a vector in $\mathbf{R}^{32^2} = \mathbf{R}^{1024}$ where each pixel value corresponds to a coefficient in the standard basis. The PCA image can be represented using just 2 coefficients corresponding to the first two PCA basis vectors.

b) If we used the standard basis, we would get the grayscale values for two of the 64^2 pixels, and the rest of the pixels would have a value of 0 (i.e., appear as pure black.) There would be no way to distinguish whether the letter is an 'a' or 'b' using just two coordinates in the standard basis.

10.6.4 Binary Classification by SVM

1)

a) The hyperplane is the line $x_1 + 2x_2 = 1$

b) The two half-spaces are the half-plane $\pi^+ : x_1 + 2x_2 > 1$ and the half-plane $\pi^- : x_1 + 2x_2 < 1$.

2)

a) For any $k \neq 0$, we have$\mathbf{w} \cdot \mathbf{x} + b = 0$ if and only if $k(\mathbf{w} \cdot \mathbf{x} + b) = 0$ if and only if $k\mathbf{w} \cdot \mathbf{x} + kb = 0$

b) The signed distance formula is $\rho(\mathbf{x}, \pi) = \frac{\mathbf{w}\cdot\mathbf{x}+b}{\|\mathbf{w}\|}$. Since $\mathbf{w} \cdot \mathbf{0} = 0$, the signed distance is $\rho(\mathbf{0}, \pi) = \frac{b}{\|\mathbf{w}\|}$.

c) As in the right panel of the 4th figure in "Intuition Underlying SVM", there are infinitely many parallel separating hyperplanes with the same margin filling in the space between support vectors. The SVM hyperplane is the one that is equidistant from the two classes C_1, C_2.

d) The SVM L_2 loss function simplifies to $J = \frac{\lambda}{2}\|\mathbf{w}\|^2$ ($\lambda > 0$), which if minimized will then maximize the margin $\frac{2}{\|\mathbf{w}\|}$.

3) See the JNB Support Vector Machines.

```
Requirement already satisfied: opencv-python in c:\users\pisihara\appdata\local\
 ↪anaconda3\lib\site-packages (4.8.0.74)
Requirement already satisfied: numpy>=1.21.2 in c:\users\pisihara\appdata\local\
 ↪anaconda3\lib\site-packages (from opencv-python) (1.24.3)
```

10.7 JNB LAB: Linear Algebra for Data Analysis

Here is an example of OLS linear regression using the datafile "ACTCollegeEligible.csv" for ACT Scores and College Eligibility.

1) Import Libraries, dropping rows with missing data.

```
df=pd.read_csv("ACTCollegeEligible.csv")
df=df.dropna()
df.head(2)
```

```
                                School Name     Street Address    ZIP  \
0                  Chicago Academy High School  3400 N Austin Ave  60634
3  William Jones College Preparatory High School     606 S State St  60605
```

(continues on next page)

(continued from previous page)

```
   Average Score ACT 2012  5-Year Cohort Graduation Rate 2012 - Percent  \
0                    19.3                                          71.0
3                    25.1                                          87.1

   One-Year Dropout Rate 2012 - Percent  \
0                                   4.0
3                                   1.5

   Freshman On-Track to Graduate 2012 - Percent  \
0                                          82.2
3                                          98.0

   College Eligibility 2012 - Percent  College Enrollment 2012 - Percent  \
0                                35.2                                75.8
3                                80.7                                92.2

   College Retention 2010 - Percent  \
0                              71.0
3                              90.8

   Misconducts Resulting in Suspensions 2012 - Percent  \
0                                                4.5
3                                               10.6

   Average Days of Suspension 2012  Student Attendance 2012 - Percent  \
0                              2.6                               93.6
3                              4.5                               93.9

   Teacher Attendance 2012 - Percent  X Coordinate  Y Coordinate  Longitude  \
0                               95.6   1135740.091   1922002.941 -87.776506
3                               95.3   1176412.354   1897618.088 -87.627755

    Latitude                     Location
0  41.942154  (-87.77650575, 41.94215439)
3  41.874419  (-87.62775497, 41.87441898)
```

2) Specify the x and y coordinates of the points to be plotted.

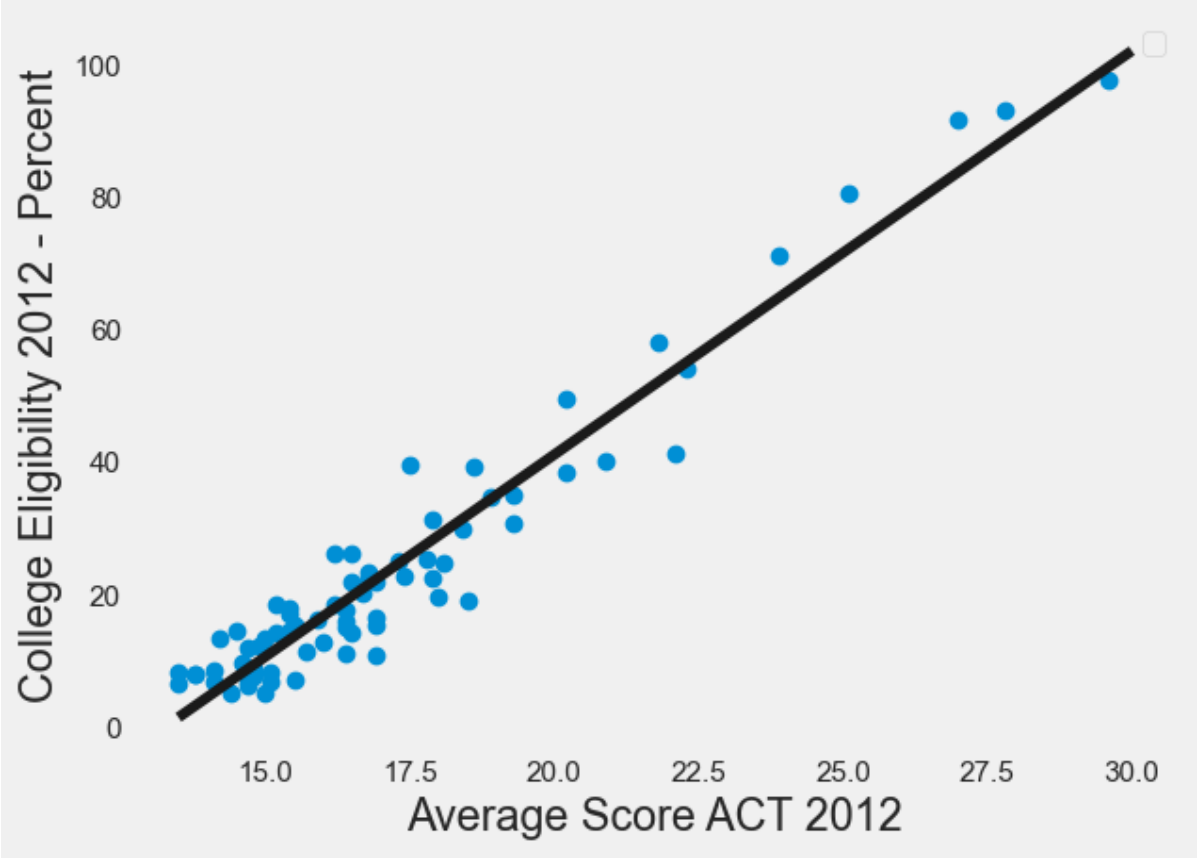
```
X=df[["Average Score ACT 2012"]]
Y=df[["College Eligibility 2012 - Percent"]]
```

3) Create the OLS regression model.

```
OLS regression line y=mx+b
slope m= [[6.09611136]]
intercept b= [-80.78259412]
```

4. Plot the data with OLS regression line.

```
Intercept is  [-80.78259412]
Slope is  [[6.09611136]]
R^2 for OLS is  0.9351396968249491
```



10.7.1 Exercise

Exercise

Execute the next cell to retrieve from the Chicago Data Portal school data. Then, for pre-K - 8 schools in Chicago ZIP Code 60623 make a scatterplot that shows the %black (x-axis) vs. %hispanic (y-axis) and include the OLS regression line on the plot.

```
raw_CPS_data=  pd.read_json('https://data.cityofchicago.org/resource/kh4r-387c.json?
↪$limit=100000')
raw_CPS_data.head(1)
```

```
   school_id  legacy_unit_id  finance_id short_name  \
0     609966            3750       23531    HAMMOND

                            long_name primary_category  is_high_school  \
0  Charles G Hammond Elementary School               ES           False

   is_middle_school  is_elementary_school  is_pre_school  ...  \
0              True                  True           True  ...

  fifth_contact_title fifth_contact_name seventh_contact_title  \
0                 NaN                NaN                   NaN
```

(continues on next page)

(continued from previous page)

```
  seventh_contact_name refugee_services visual_impairments  \
0                  NaN              NaN                NaN

  freshman_start_end_time sixth_contact_title  sixth_contact_name  \
0                     NaN                NaN                 NaN

   hard_of_hearing
0              NaN

[1 rows x 92 columns]
```

10.7.2 K-Means Clustering

In this section, we will illustrate how to cluster a larger number of points in analyzing Chicago violent crime data in the file "Violence.csv"

Data source: Chicago Data Portal (https://data.cityofchicago.org/Public-Safety/Violence-Reduction-Victims-of-Homicides-and-Non-Fa/gumc-mgzr). Data was extracted on December 7th, 2022.

1) Read in recent data for 1,000 violent crime occurrences specified by the latitude-longitude location of each incident.

```
Size of Dataset (1000, 2)
```

```
    LATITUDE  LONGITUDE
0  41.865451  -87.72505
```

2) Next, we map the 1,000 recorded instances to see what we are working with.

Note

Please visit the web version of this section to engage with this interactive resource.

```
<folium.folium.Map at 0x139f4864190>
```

3) We've provided a function that will allow you to test different numbers of clusters for the same data. For the purposes of this lab, up to 100 clusters can be used. We will use this function to visualize 22 clusters since there are 22 police districts in Chicago.

4) Let's use make_map() to divide the locations into two parts.

Note

Please visit the web version of this section to engage with this interactive resource.

```
make_map(2,violence)
```

```
C:\Users\pisihara\AppData\Local\anaconda3\Lib\site-packages\sklearn\cluster\_
↪kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10␣
↪to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
```

(continues on next page)

(continued from previous page)

```
  warnings.warn(
C:\Users\pisihara\AppData\Local\anaconda3\Lib\site-packages\sklearn\cluster\_
 ↪kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows␣
 ↪with MKL, when there are less chunks than available threads. You can avoid it by␣
 ↪setting the environment variable OMP_NUM_THREADS=4.
  warnings.warn(
```

```
<folium.folium.Map at 0x139f7480dd0>
```

Note that the I55 Expressway which runs northeast from Summit to Chicago is an approximate dividing line between the clusters on the South and West sides of Chicago.

Assignment

Assignment

Use k-means to divide the crime data into 22 clusters. How does the result compare with a map of the 22 Chicago Police Department districts?

10.7.3 PCA

1) The following function converts images in a folder to a matrix of column vectors containing pixel values for each image.
2) Run the next code cells to test if you were able to use the correct path name. If not, go back and check if the images are in the file you copied the path name of, or check if you copied the correct path name.

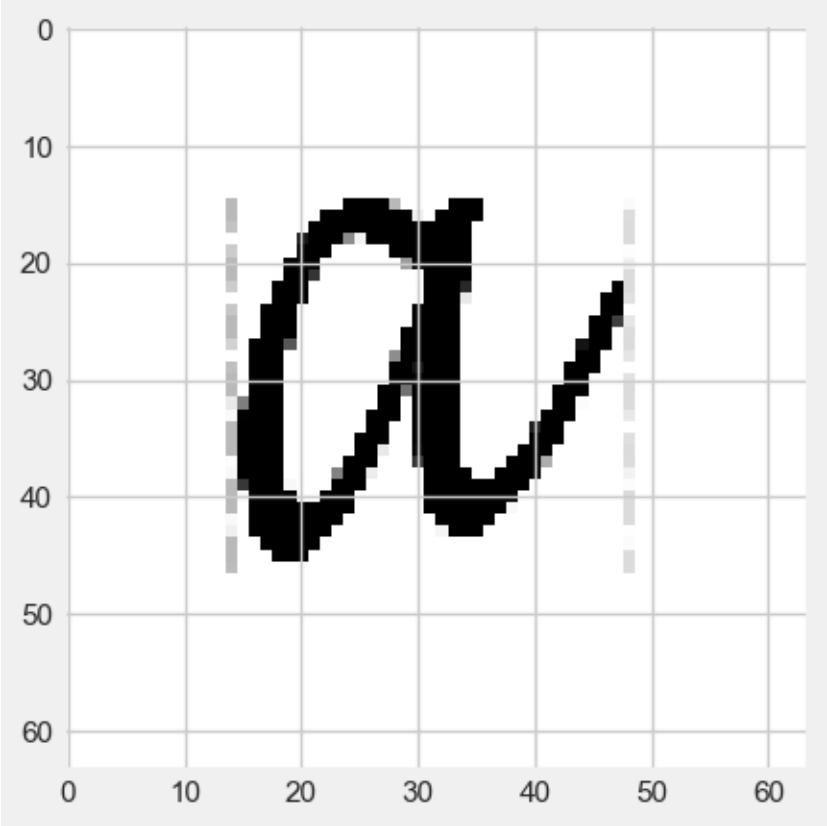
- The data set will be the 8 images in the folder "letters" with a 64×64 resolution.

```
[traindata,imgs]=imagetovector(64,"letters",8)
traindata.head(4)
```

	0	1	2	3	4	5	6	7
0	255.0	255.0	255.0	255.0	255.0	255.0	255.0	255.0
1	255.0	255.0	255.0	255.0	255.0	255.0	255.0	255.0
2	255.0	255.0	255.0	255.0	255.0	255.0	255.0	255.0
3	255.0	255.0	255.0	255.0	255.0	255.0	255.0	255.0

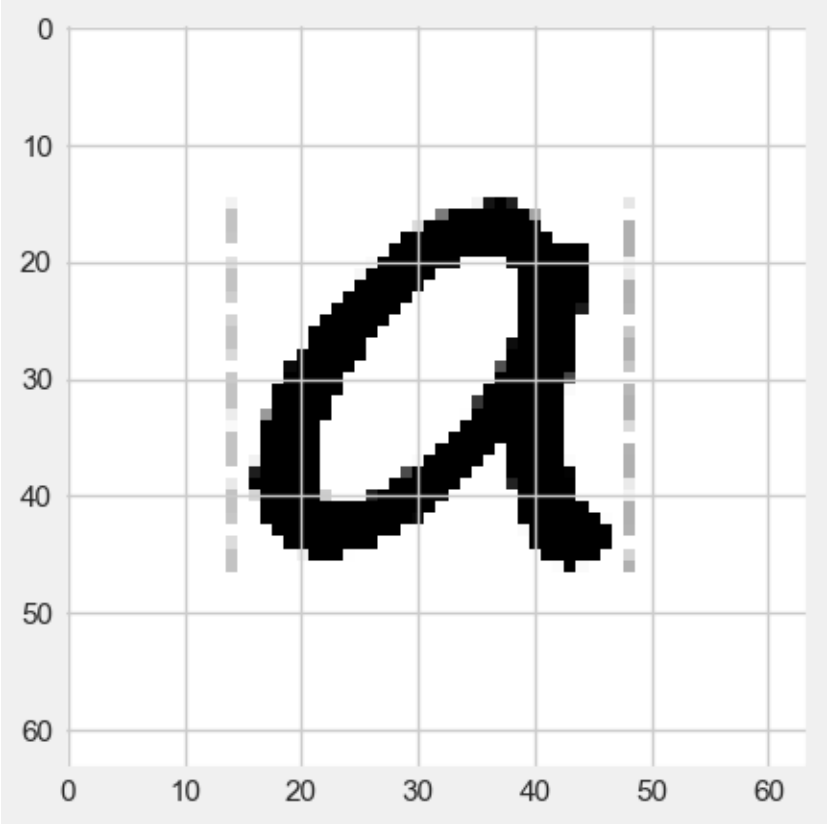
```
#display the first  image
plt.imshow(imgs[0], cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x139f7232a10>
```



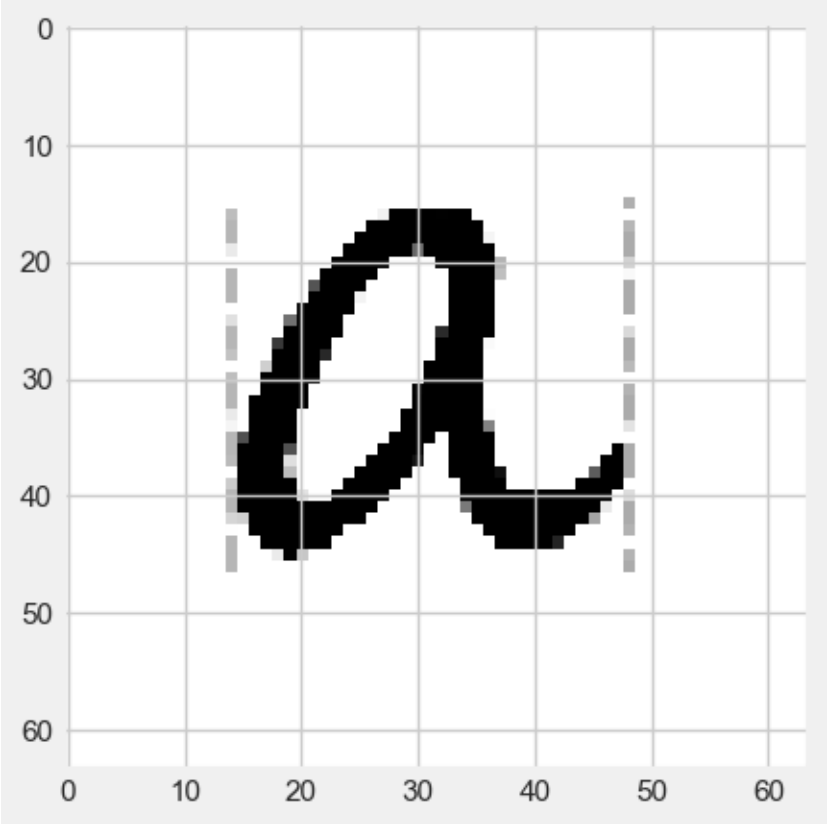

```
#display the second  image
plt.imshow(imgs[1], cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x139f800e490>
```



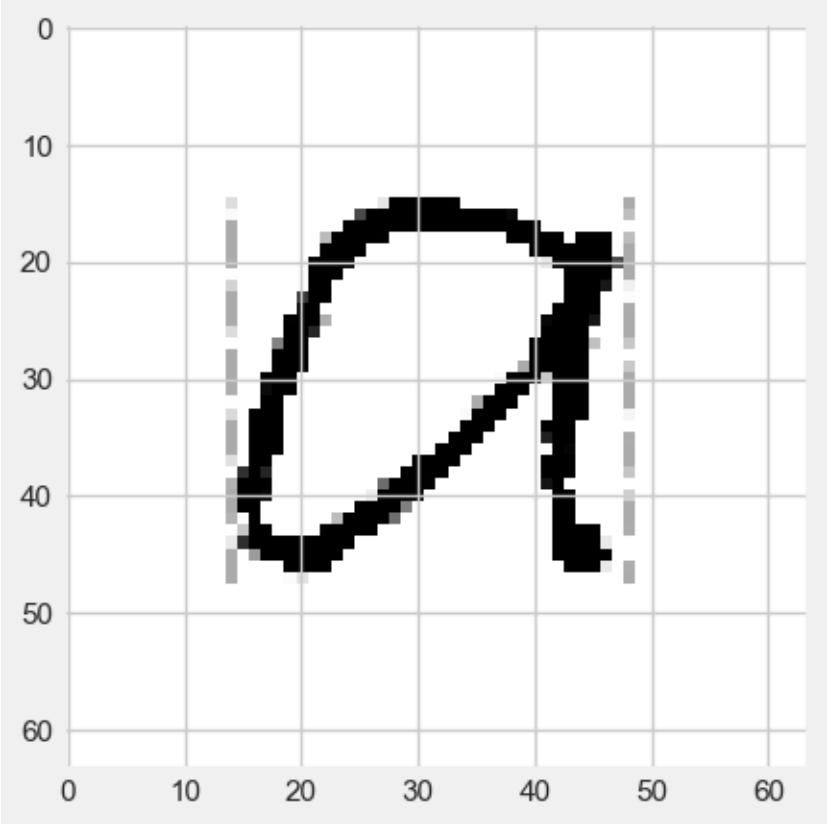

```
#display the third image
plt.imshow(imgs[2], cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x139f76bc910>
```



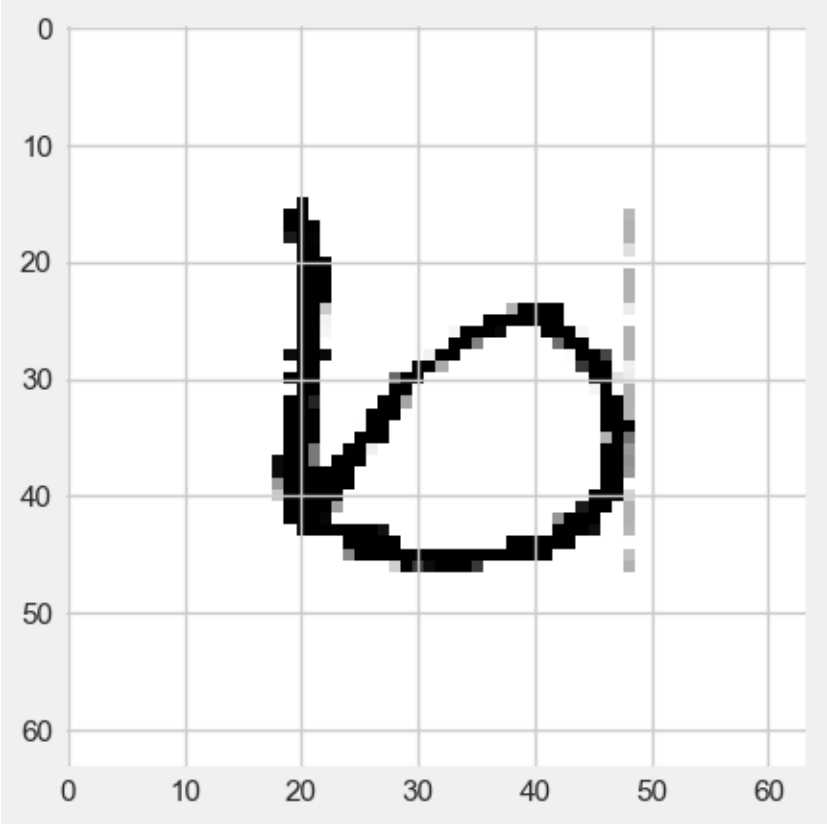

```
#display the fourth training image
plt.imshow(imgs[3], cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x139f7716490>
```



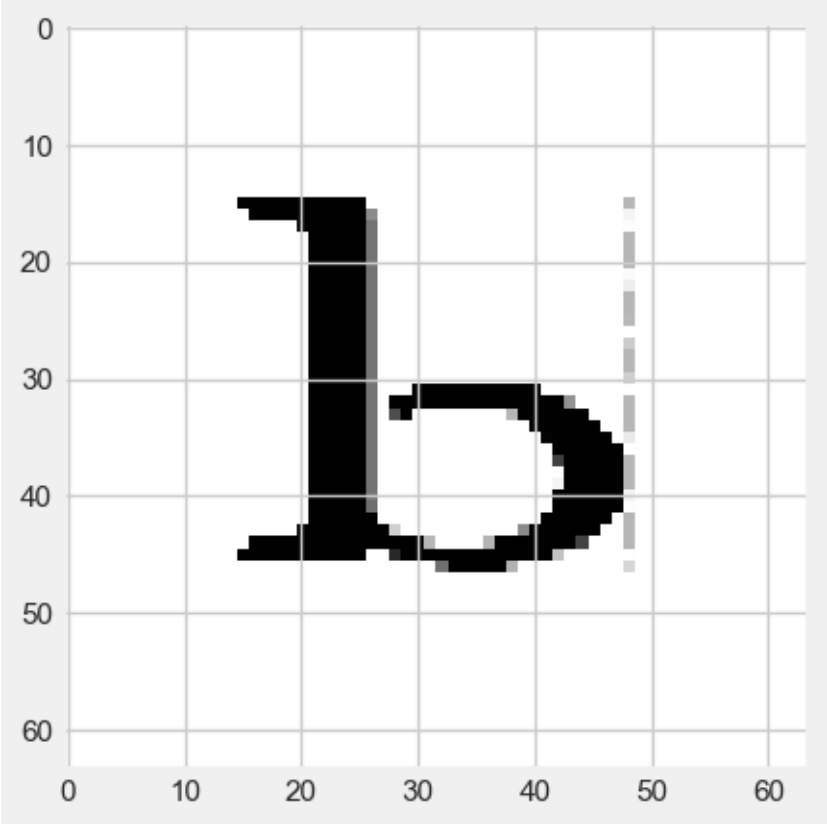

```
#display the fifth training image
plt.imshow(imgs[4], cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x139f78b7250>
```



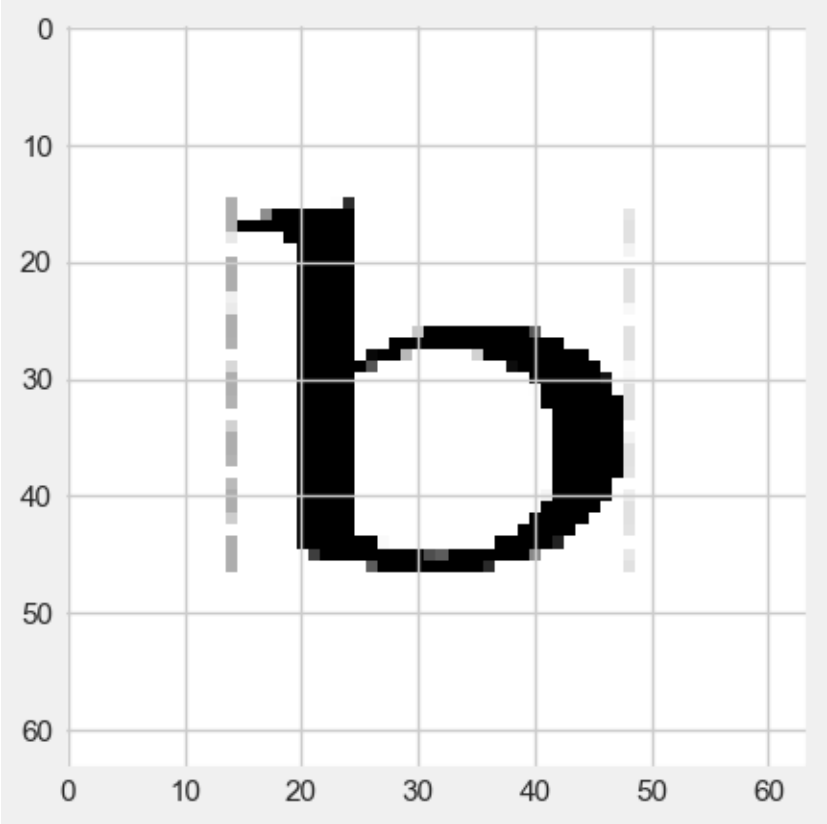

```
#display the sixth image
plt.imshow(imgs[5], cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x139f792b5d0>
```



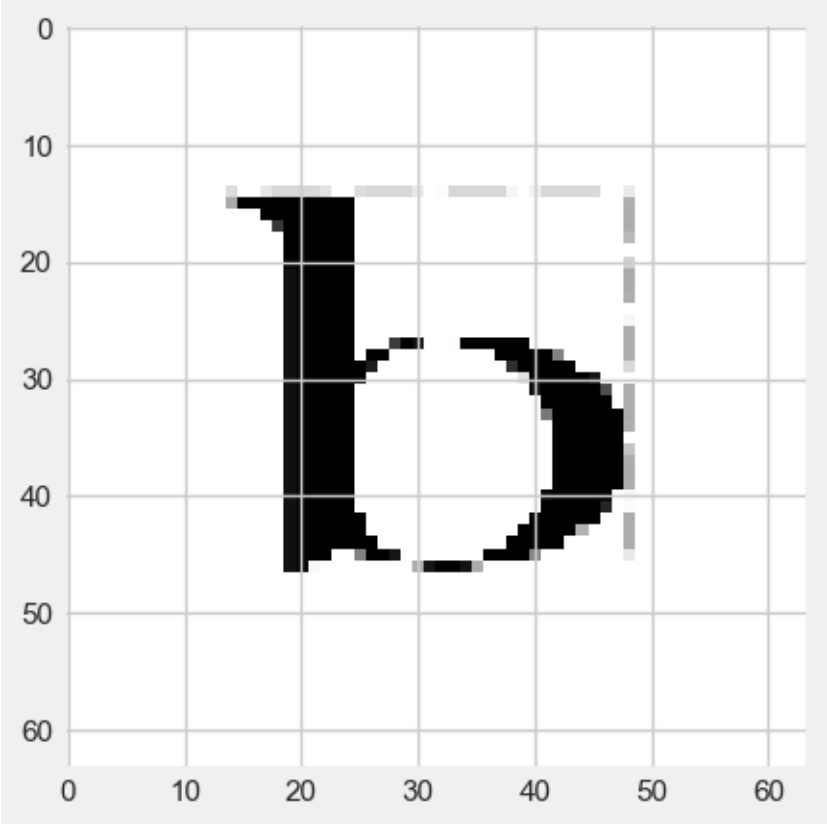

```
#display the seventh image
plt.imshow(imgs[6], cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x139f794a590>
```




```
#display the eight image
plt.imshow(imgs[7], cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x139f77b0ad0>
```




```
#store the data in an Excel file
letter=traindata
letter.to_excel("ab.xlsx")
```

3) We will now use PCA to project the data onto the first 2 principal component vectors.

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(np.transpose(letter))
letter_pca = pca.transform(np.transpose(letter))
print("original shape:   ", np.transpose(letter).shape)
print("transformed shape:", letter_pca.shape)
```

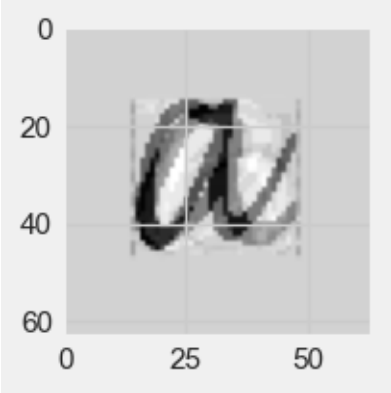
```
original shape:    (8, 4096)
transformed shape: (8, 2)
```

4) Let's check how well we can recognize the 2-dimensional versions of the first image.

```
filtered = pca.inverse_transform(letter_pca)
```

```
fig=plt.figure(figsize=(2,2))
plt.gca().imshow(filtered[0].reshape(64, 64),
                 cmap="gray")
```

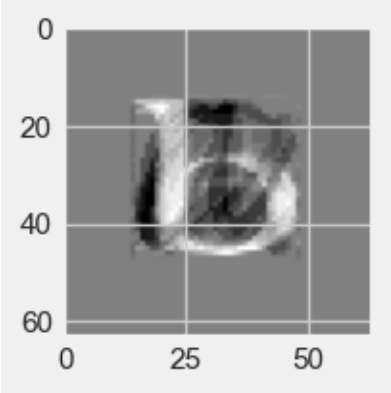
```
<matplotlib.image.AxesImage at 0x139f7179350>
```



5) Here is the image of the first principal component vector.

```
fig=plt.figure(figsize=(2,2))
plt.gca().imshow(pca.components_[0].reshape(64, 64),
                                    cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x139f6df6490>
```



Exercise

Exercise

a) Display the two-dimensional version of the 5th image (the first letter b)

b) Display the image corresponding to the first and the second principal component vectors.

c) Give the components of the image in part a) in terms of the vectors in part b)

d) What would happen if instead of using the PCA basis, we used two standard basis vectors for the projections (eg. $e_1 = (1, 0, 0, 0, ..., 0)$ and $e_2 = (0, 1, 0, 0, ..., 0)$?

10.7.4 Support Vector Machines

Given a data set with each point labeled by membership in one of two classes, support vector machines (SVMs) can be used to create a classification of all such data using a linear boundary.

Here we will explore Chicago housing data by ZIP Code. In particular, we will look at the median monthly rent in 2016 and the proportion of single parents in 2016.

Datafile: Housing.xlsx

Source: American Community Survey

1) Let's read in the data.

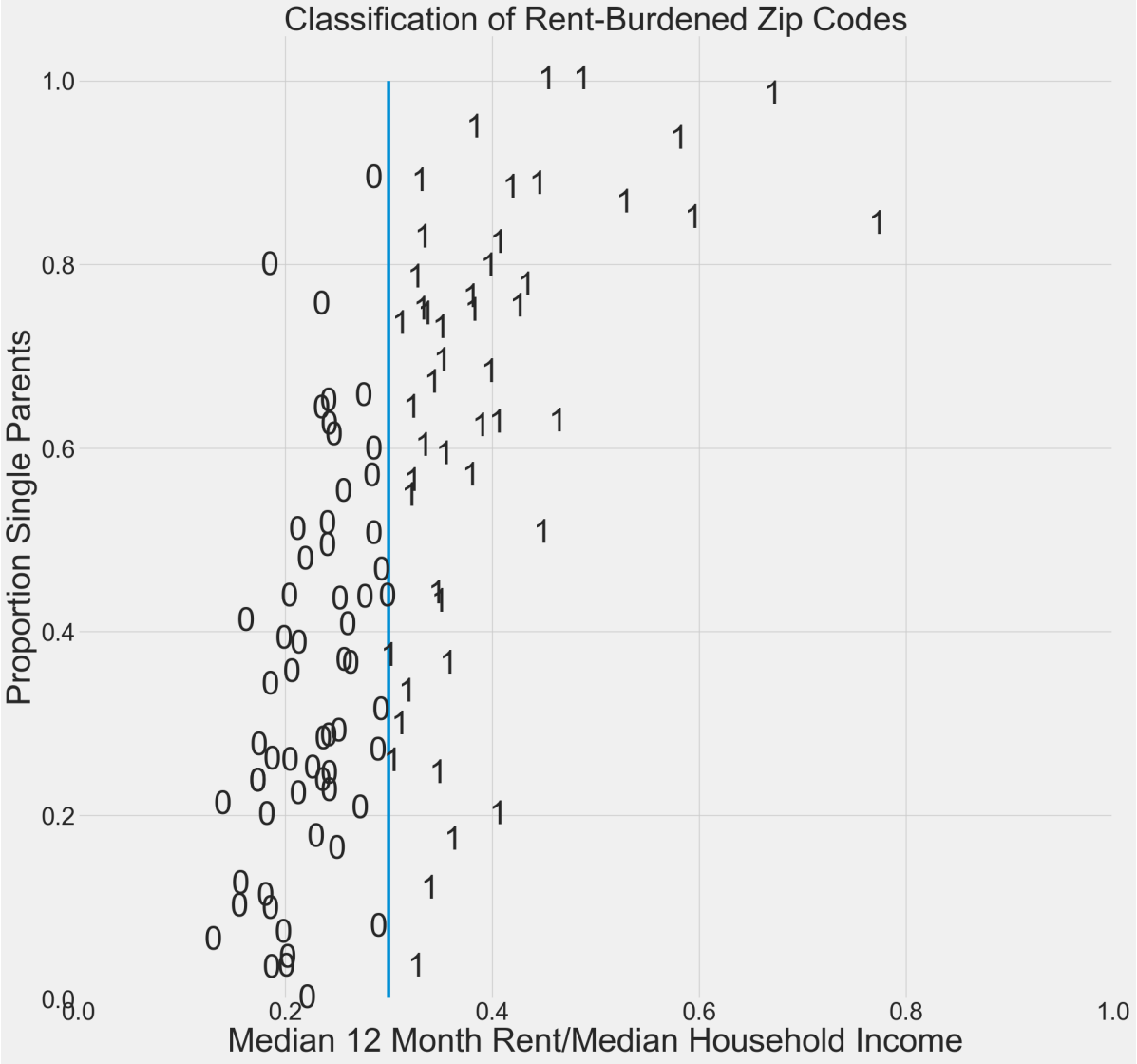
```
housing = pd.read_excel('Housing.xlsx')
housing.head(2)
```

```
   Unnamed: 0  rent  singleparent  income
0           0   815        0.2362   41380
1           1   905        0.3673   42253
```

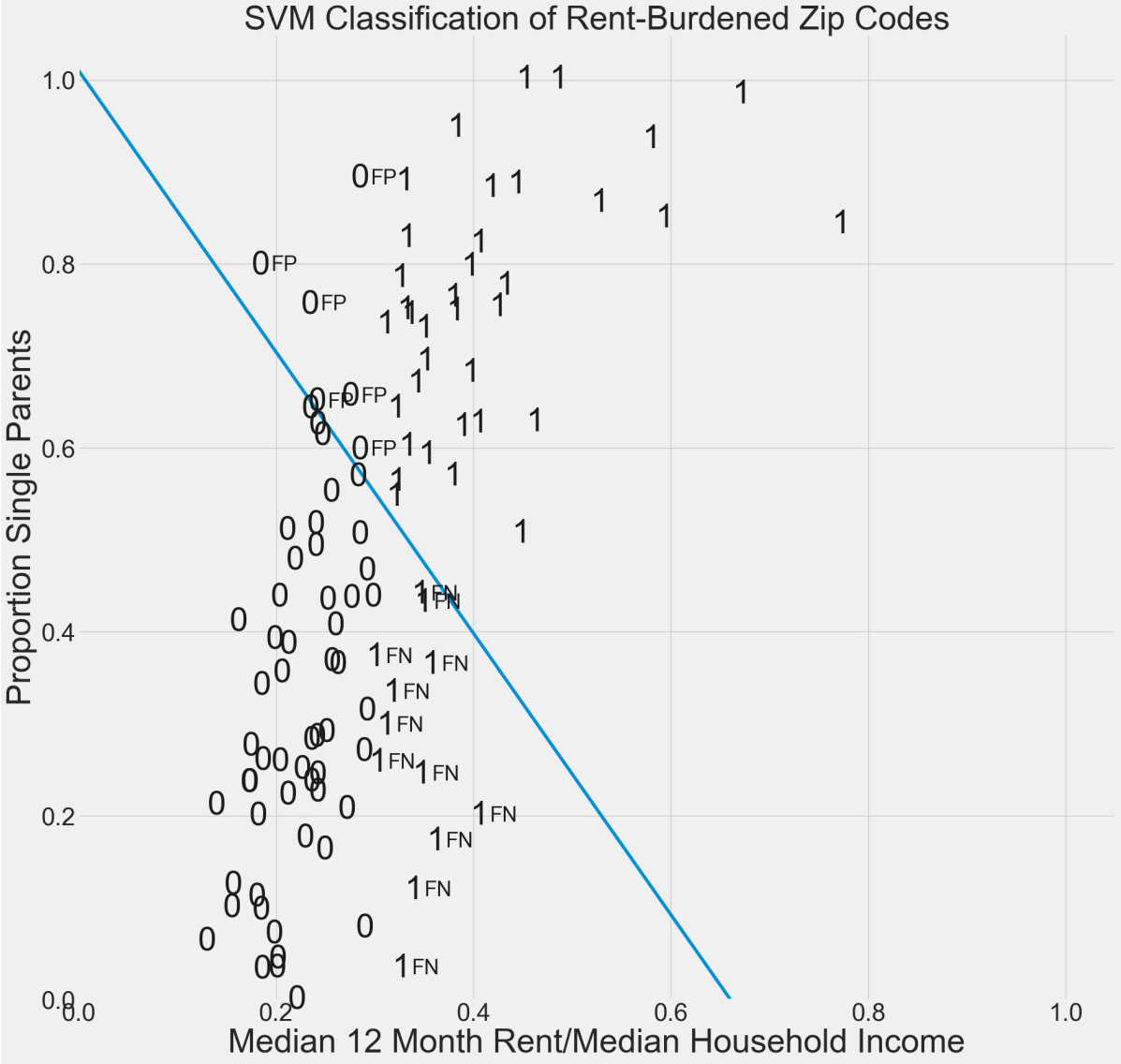
2) Let's create a column "rent burden" which gives the proportion of income spent on rent. We will label households spending less than 30% as class 0 (not rent-burdened) and the remainder in class 1 (rent-burdened).

```
   Unnamed: 0  rent  singleparent  income  rent burden  class
0           0   815        0.2362   41380     0.236346    0.0
1           1   905        0.3673   42253     0.257023    0.0
2           2   677        0.4372   39784     0.204203    0.0
3           3  1019        0.0437   60510     0.202082    0.0
4           4  1306        0.4110   96576     0.162276    0.0
5           5  1617        0.0631  148825     0.130381    0.0
6           6   980        0.2852   48652     0.241717    0.0
7           7   921        0.3333   34741     0.318126    1.0
8           8   764        0.5053   32095     0.285652    0.0
9           9   939        0.8277   33805     0.333323    1.0
```

3) Let's make a graph of "x = rent burden" and "y = singleparent"



4) Now let's use support vector machine classification to read in the data points and to find a hyperplane (straight line) that is used as the basis of classification.



Note that the SVM hyperplane does not correctly classify the labeled data. Find formulas for precision and recall and give these misclassification scores. Also give the F-score, which is the geometric mean of recall and precision.

```
TP= 36
FP= 6
FN= 12
TN= 6
Accuracy= 0.7
Recall= 0.75
Precision= 0.8571428571428571
F-score= 0.7999999999999999
```

By increasing the value of the regularization parameter C, we can reduce misclassifications.

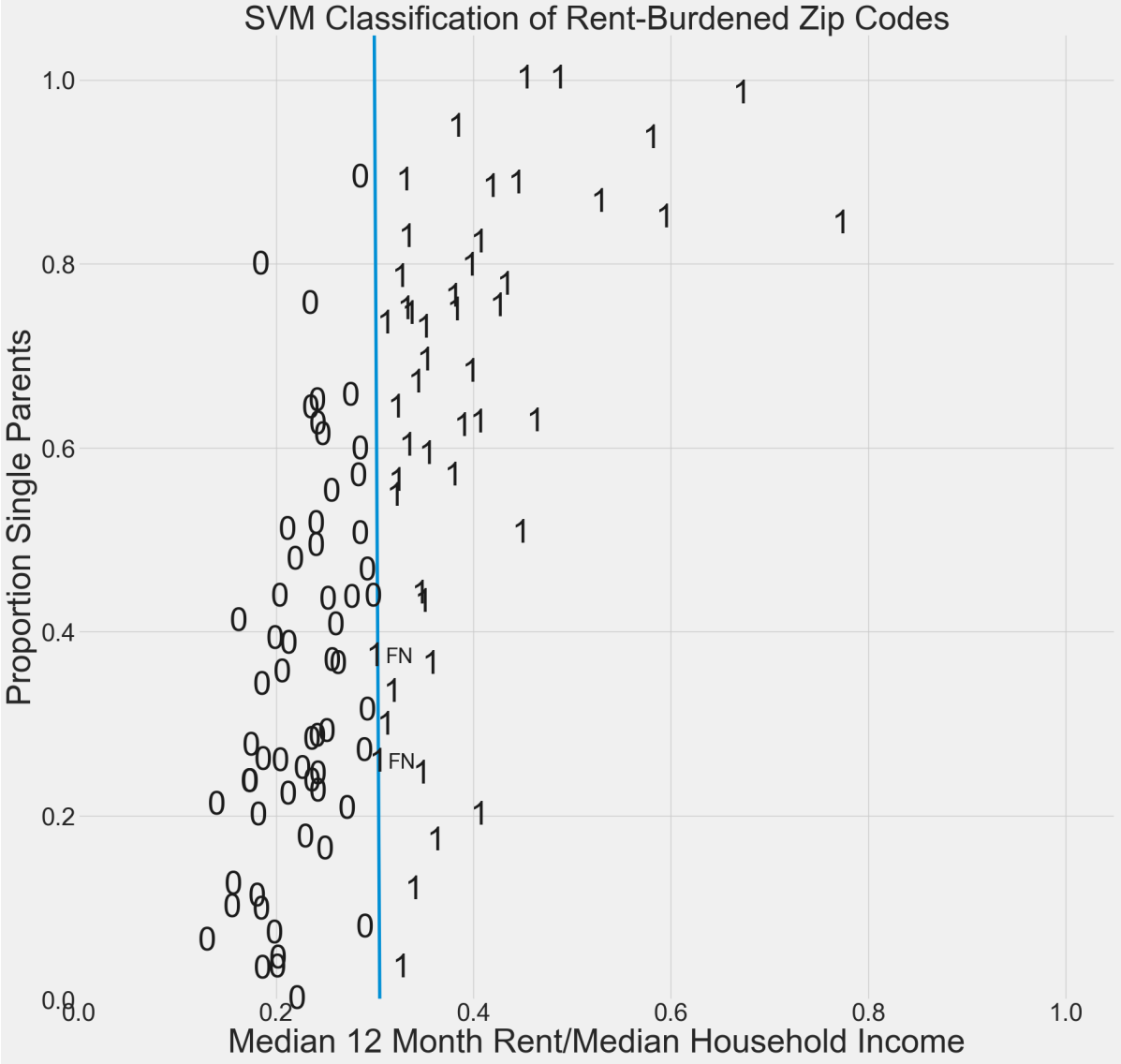


Image Recognition

We will now show how SVM can be used in image recognition.

Download the images folder and put the 'images' sub-folder in the same folder as this notebook.

Find the pathname of the file on your hard drive, copy it to your clipboard, and paste it in the quotations of the second argument below. - Instructions on gathering the pathname for Windows operating systems can be found here, and for Mac operating systems can be found here.

Run the next three code cells to test if you were able to use the correct path name. If not, go back and check if the images are in the file you copied the path name of, or check if you copied the correct path name.

- The training set will be the 6 images in the folder "images" with a 32×32 resolution.

```
[traindata,imgs]=imagetovector(32,"images",6)
traindata.head(2)
```

```
      0      1     2      3     4     5
0  92.0  108.0  65.0  176.0  21.0  86.0
1  82.0  111.0  65.0  182.0  28.0  84.0
```

Run the cell below to train a linear support vector machine on the training images. This is building a hyperplane in an $n = 1024$ dimensional space to separate the data.

- Notice how the `Y` variable is the vector of classes for each image. This is because 0 corresponds to Mayor Daley, and 1 corresponds to Mayor Washington.

```
model = SVC(kernel='linear', C=1)
X=[traindata.loc[:,"0"],traindata.loc[:,"1"],traindata.loc[:,"2"],traindata.loc[:,"3
 ↪"],traindata.loc[:,"4"],traindata.loc[:,"5"]]
Y=[0,0,0,1,1,1] #Labels the images 0=Mayor Daley 1=Mayor Washington
model.fit(X,Y)
ypred=model.predict(X)
ypred
```

```
array([0, 0, 0, 1, 1, 1])
```

Ideally, an SVM will perfectly classify the exact data it was trained on. This model does happen to correctly identify which mayor is which. Check to see if the `Y` vector matches the `y_pred` vector.

Download the imagestestingset folder from the Google Drive onto your hard drive. Place the folder in the same directory as this notebook and the other training image folder.

Copy the folder path name and paste it into the second argument of the `imagetovector` function below, denoted `__CHANGE__THIS__TO__TRAINING__IMAGE__PATHNAME__`.

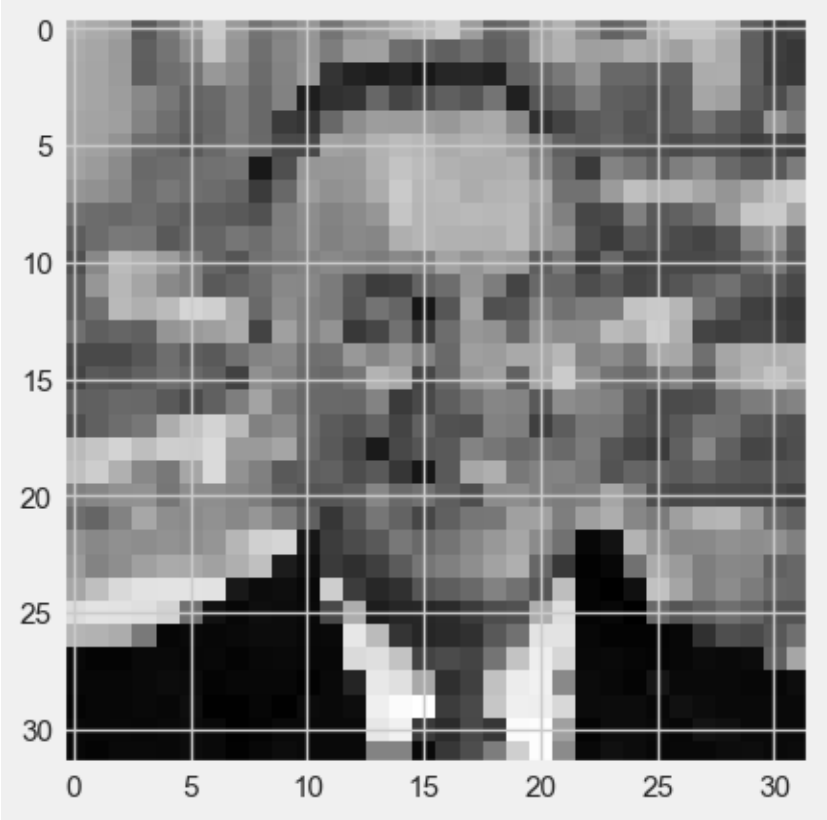
- The test set will be the 4 images in the folder "testimages" with a 32×32 resolution.

```
[testdata,testimgs]=imagetovector(32,"testimages",4)
testdata.tail(2)
```

```
         0     1     2      3
1022  18.0  55.0  47.0   13.0
1023  19.0  48.0  52.0  169.0
```

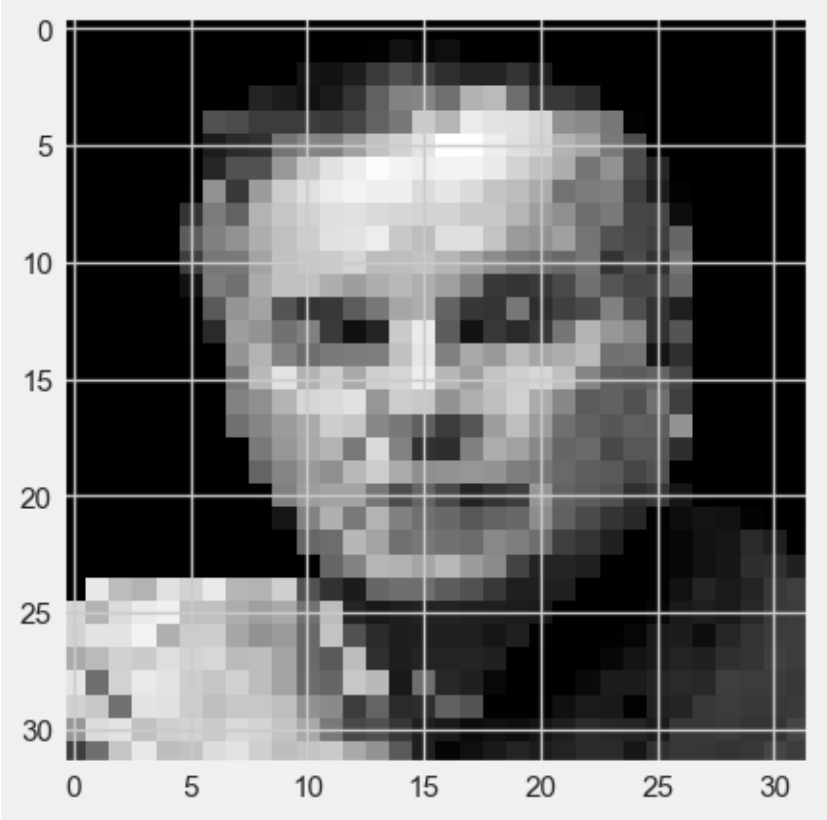
```
#display the first test image
plt.imshow(testimgs[0], cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x139f6acc910>
```



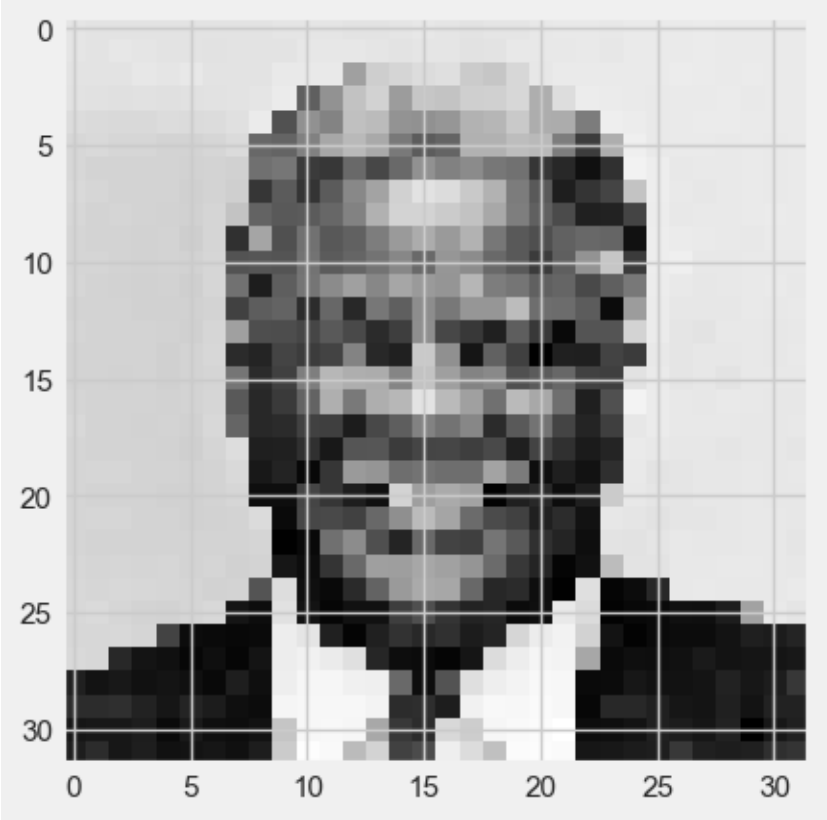

```
#display the second test image
plt.imshow(testimgs[1], cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x139fba1fb50>
```



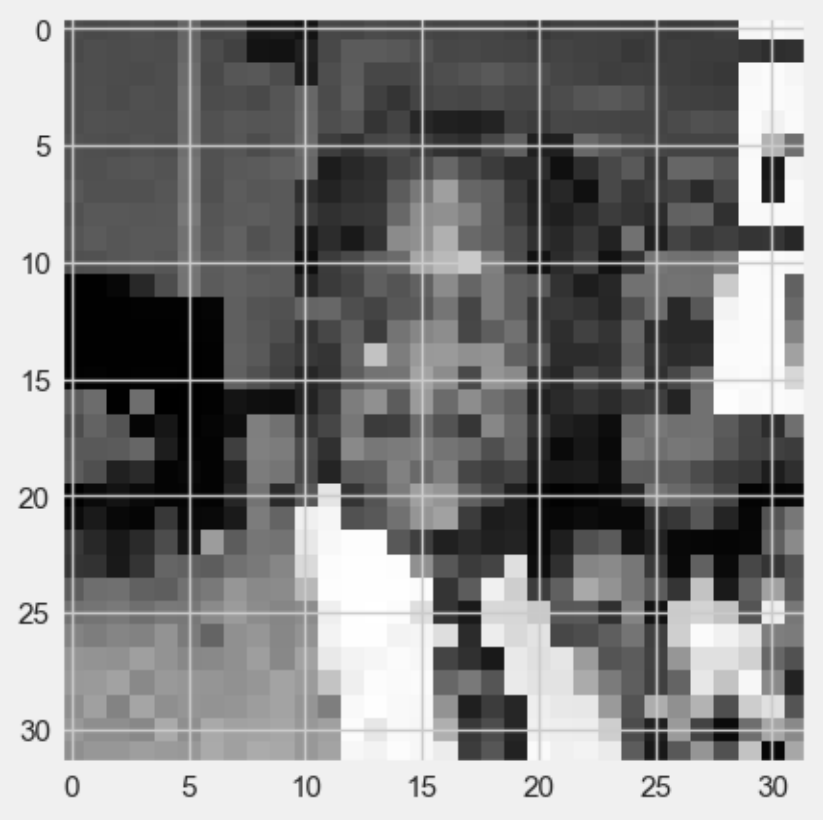

```
#display the third test image
plt.imshow(testimgs[2], cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x139fba46490>
```




```
#display the fourth test image
plt.imshow(testimgs[3], cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x139fbc40390>
```



Run the two lines in the cell below to see how the SVM classifies the 4 new images in the folder "testimages" (the values should be $[0, 0, 1, 1]$ since the first two images are Mayor Daley and the second two are Mayor Washingon)

```
Xtest=[testdata.loc[:,"0"],testdata.loc[:,"1"],testdata.loc[:,"2"],testdata.loc[:,"3
 ↪"]]
model.predict(Xtest)
```

```
array([1, 0, 1, 1])
```

(Note: An output of [1,0,1,1], for example, would indicate that the SVM misclassifies the first image as Mayor Washington, and gets the other three images correct.)

Exercise

Repeat the above analysis using images of Michelle Obama and Hillary Clinton (both First Ladies were raised in Chicago).

```
Requirement already satisfied: opencv-python in c:\users\pisihara\appdata\local\
 ↪anaconda3\lib\site-packages (4.8.0.74)
Requirement already satisfied: numpy>=1.21.2 in c:\users\pisihara\appdata\local\
 ↪anaconda3\lib\site-packages (from opencv-python) (1.24.3)
```

10.8 JNB Lab Solutions

10.8.1 Section 1 Exercise

```
   school_id  legacy_unit_id  finance_id short_name  \
0     609966            3750       23531    HAMMOND

                            long_name primary_category  is_high_school  \
0  Charles G Hammond Elementary School               ES           False

   is_middle_school  is_elementary_school  is_pre_school  ...  \
0              True                  True           True  ...

  fifth_contact_title fifth_contact_name seventh_contact_title  \
0                 NaN                NaN                   NaN

  seventh_contact_name refugee_services visual_impairments  \
0                  NaN              NaN                NaN

  freshman_start_end_time sixth_contact_title  sixth_contact_name  \
0                     NaN                NaN                 NaN

   hard_of_hearing
0              NaN

[1 rows x 92 columns]
```

```
raw_CPS_data.columns
```

```
Index(['school_id', 'legacy_unit_id', 'finance_id', 'short_name', 'long_name',
       'primary_category', 'is_high_school', 'is_middle_school',
       'is_elementary_school', 'is_pre_school', 'summary',
       'administrator_title', 'administrator', 'secondary_contact_title',
       'secondary_contact', 'address', 'city', 'state', 'zip', 'phone', 'fax',
       'cps_school_profile', 'website', 'facebook', 'attendance_boundaries',
       'grades_offered_all', 'grades_offered', 'student_count_total',
       'student_count_low_income', 'student_count_special_ed',
       'student_count_english_learners', 'student_count_black',
       'student_count_hispanic', 'student_count_white', 'student_count_asian',
       'student_count_native_american', 'student_count_other_ethnicity',
       'student_count_asian_pacific', 'student_count_multi',
       'student_count_hawaiian_pacific', 'student_count_ethnicity_not',
       'statistics_description', 'demographic_description', 'dress_code',
       'prek_school_day', 'kindergarten_school_day', 'school_hours',
       'after_school_hours', 'earliest_drop_off_time', 'classroom_languages',
       'bilingual_services', 'title_1_eligible', 'preschool_inclusive',
       'preschool_instructional', 'transportation_bus', 'transportation_el',
       'school_latitude', 'school_longitude', 'overall_rating',
       'rating_status', 'rating_statement', 'classification_description',
       'school_year', 'third_contact_title', 'third_contact_name', 'network',
       'is_gocps_participant', 'is_gocps_prek', 'is_gocps_elementary',
       'is_gocps_high_school', 'open_for_enrollment_date', 'twitter',
       'youtube', 'pinterest', 'college_enrollment_rate_school',
       'college_enrollment_rate_mean', 'graduation_rate_school',
       'graduation_rate_mean', 'significantly_modified',
       'transportation_metra', 'fourth_contact_title', 'fourth_contact_name',
```

(continues on next page)

(continued from previous page)

```
       'fifth_contact_title', 'fifth_contact_name', 'seventh_contact_title',
       'seventh_contact_name', 'refugee_services', 'visual_impairments',
       'freshman_start_end_time', 'sixth_contact_title', 'sixth_contact_name',
       'hard_of_hearing'],
      dtype='object')
```

```
raw_CPS_data['grades_offered'].value_counts()
```

```
PK,K-8       327
9-12         144
K-8           82
7-12          11
PK,K-6        10
PK,K-5        10
6-12           9
K-6            8
6-8            6
PK,K-4         4
K-12           4
PE,PK,K-8      4
11-12          4
5-8            3
PK             3
K-5            3
PK,K-3         3
8-12           2
PK,K-2         2
7-8            2
PK,3-8         1
9              1
K,4-8          1
K-1,5-8        1
3-12           1
K-3,5-8        1
1-8            1
PK,K-7         1
10-12          1
4-11           1
K-3            1
K-2            1
4-8            1
Name: grades_offered, dtype: int64
```

```
df=raw_CPS_data[['address','student_count_total','student_count_black','student_count_
↪hispanic','student_count_white','zip']]
df23=df[df['zip']==60623]
df23=df23.reset_index(drop=True)
df23.columns= ["address","total","black","hispanic","white","zip"]
df23.head(1)
```

	address	total	black	hispanic	white	zip
0	2819 W 21ST PL	342	33	304	2	60623

```
for i in df23.index:
    df23.loc[i,'%black']=round(100*df23.loc[i,'black']/df23.loc[i,'total'],1)
```

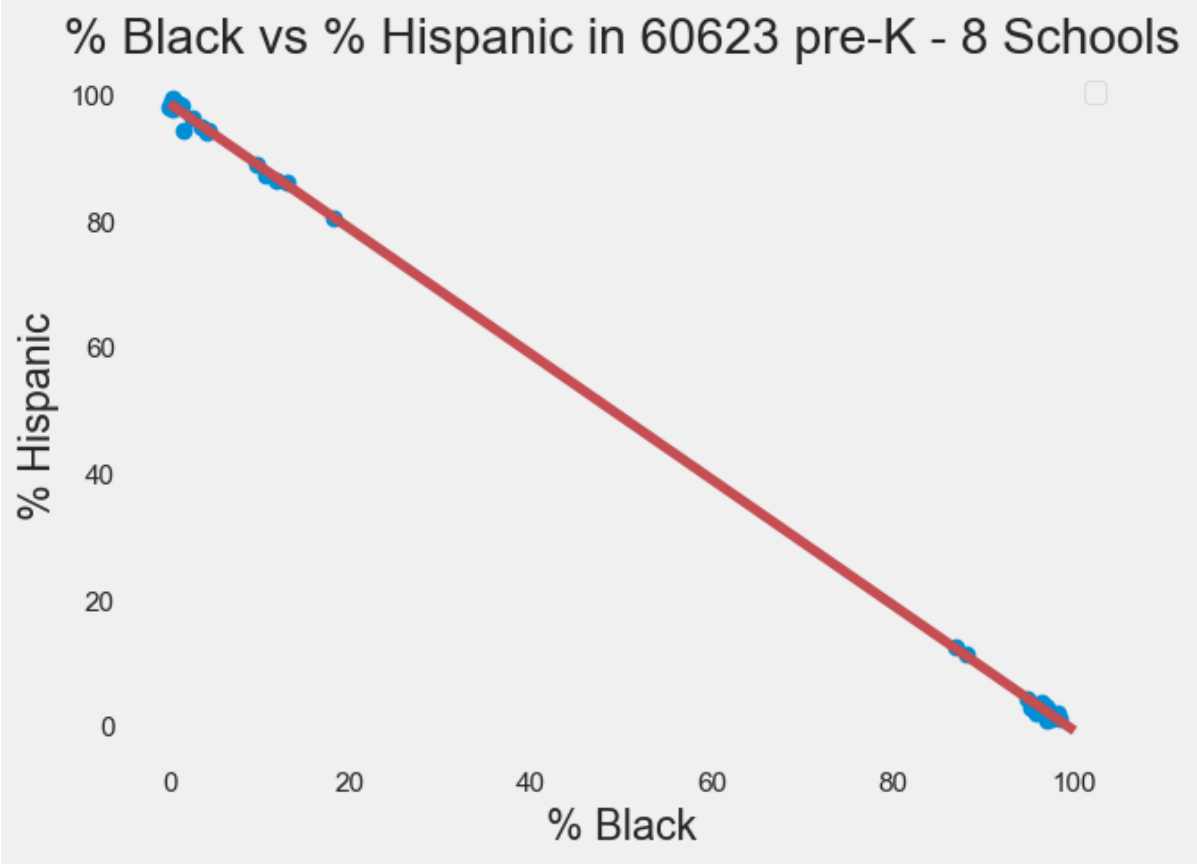
(continues on next page)

(continued from previous page)

```
    df23.loc[i,'%hispanic']=round(100*df23.loc[i,'hispanic']/df23.loc[i,'total'],1)
    df23.loc[i,'%white']=round(df23.loc[i,'white']/df23.loc[i,'total'],1)
df23.head(1)
```

```
          address  total  black  hispanic  white    zip  %black  %hispanic  \
0  2819 W 21ST PL    342     33       304      2  60623     9.6       88.9

   %white
0     0.0
```

```
Intercept is  [98.71284906]
Slope is  [[-0.99515441]]
R^2 for OLS is  0.9996943205528053
```



10.8.2 Section 2 Exercise

Let's take a look at the 22 clusters.

Note

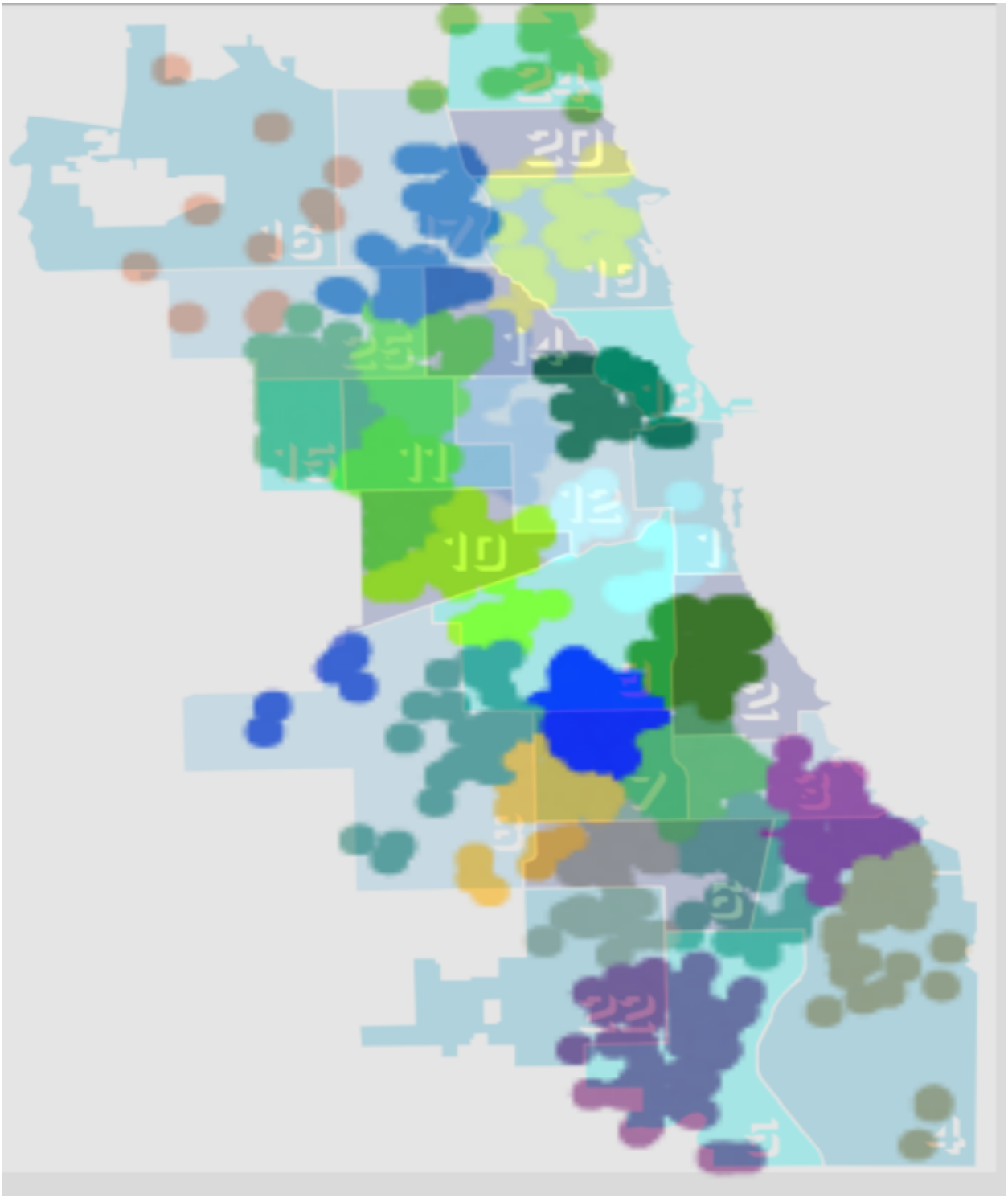
Please visit the web version of this section to engage with this interactive resource.

```
cluster22_map = make_map(22,violence)
cluster22_map
```

```
C:\Users\pisihara\AppData\Local\anaconda3\Lib\site-packages\sklearn\cluster\_
↪kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10␣
↪to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\pisihara\AppData\Local\anaconda3\Lib\site-packages\sklearn\cluster\_
↪kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows␣
↪with MKL, when there are less chunks than available threads. You can avoid it by␣
↪setting the environment variable OMP_NUM_THREADS=4.
  warnings.warn(
```

```
<folium.folium.Map at 0x1d8fa45d610>
```

Here is an overlay of this map onto police districts.



While it is not perfect, our k-means algorithm clustered the violent occurrences in a similar manner to the police district boundaries.

10.8.3 Section 3 Exercise

```
[traindata,imgs]=imagetovector(64,"letters",8)
traindata.head(4)
```

	0	1	2	3	4	5	6	7
0	255.0	255.0	255.0	255.0	255.0	255.0	255.0	255.0
1	255.0	255.0	255.0	255.0	255.0	255.0	255.0	255.0
2	255.0	255.0	255.0	255.0	255.0	255.0	255.0	255.0
3	255.0	255.0	255.0	255.0	255.0	255.0	255.0	255.0

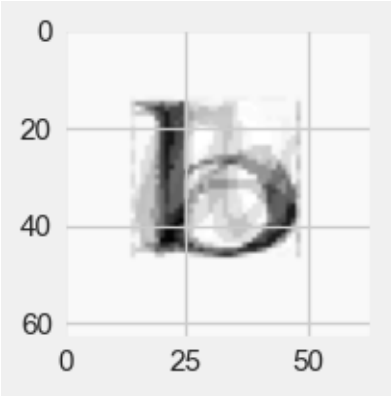
```
from sklearn.decomposition import PCA
letter=traindata
pca = PCA(n_components=2)
pca.fit(np.transpose(letter))
letter_pca = pca.transform(np.transpose(letter))
filtered = pca.inverse_transform(letter_pca)
print("original shape:   ", np.transpose(letter).shape)
print("transformed shape:", letter_pca.shape)
```

```
original shape:    (8, 4096)
transformed shape: (8, 2)
```

Answer to part a)

```
fig=plt.figure(figsize=(2,2))
plt.gca().imshow(filtered[4].reshape(64, 64),
                                 cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x1d880976490>
```



Answer to part b)

Image of first principal component vector

```
#image corresponding to the 1st basis vector.
fig=plt.figure(figsize=(2,2))
plt.gca().imshow(pca.components_[0].reshape(64, 64),
                                 cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x1d880931610>
```

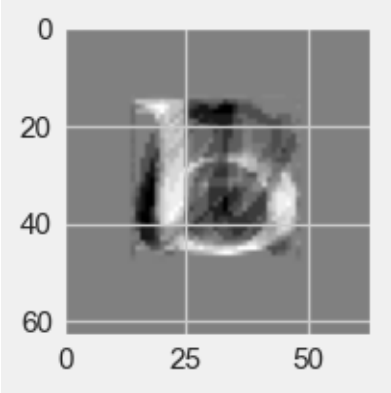
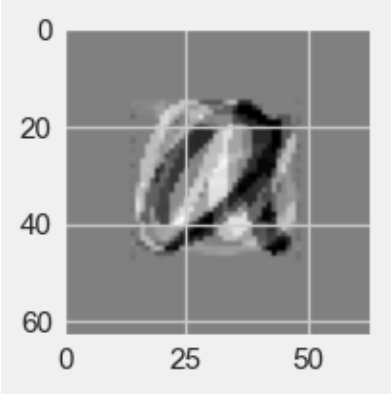


Image of the second principal component vector.

```
#image corresponding to the 2nd basis vector.
fig=plt.figure(figsize=(2,2))
plt.gca().imshow(pca.components_[1].reshape(64, 64),
                                    cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x1d8809d6490>
```



Answer to c).

```
letter_pca[1]
```

```
array([1286.48056884, 3118.77874561])
```

Answer to d) We could only model the first two pixels so would not have any idea of the images.

10.8.4 Section 4 Exercise

```
[traindata,imgs]=imagetovector(32,"exerciseimages",8)
traindata.head(2)
```

```
      0     1    2     3      4      5     6      7
0  54.0  29.0  0.0  17.0  215.0  245.0  88.0  197.0
1  41.0  65.0  0.0  16.0  182.0  242.0  93.0  186.0
```

```
model = SVC(kernel='linear', C=1)
X=[traindata.loc[:,"0"],traindata.loc[:,"1"],traindata.loc[:,"2"],traindata.loc[:,"3
 ↪"],traindata.loc[:,"4"],traindata.loc[:,"5"],traindata.loc[:,"6"],traindata.loc[:,"7
 ↪"]]
Y=[0,0,0,0,1,1,1,1] #Labels the images 0=Hillary Clinton 1=Michelle Obama
model.fit(X,Y)
ypred=model.predict(X)
ypred
```

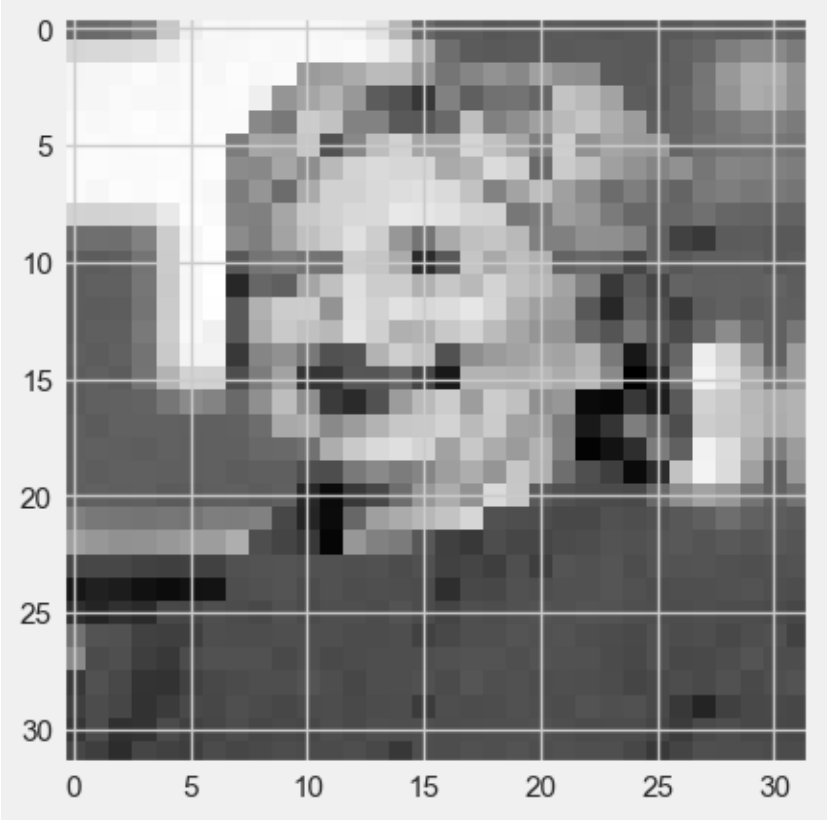
```
array([0, 0, 0, 0, 1, 1, 1, 1])
```

```
[testdata,testimgs]=imagetovector(32,"exercisetestimages",4)
testdata.tail(2)
```

```
          0     1     2     3
1022  80.0  69.0  36.0  42.0
1023  80.0  44.0  37.0  47.0
```

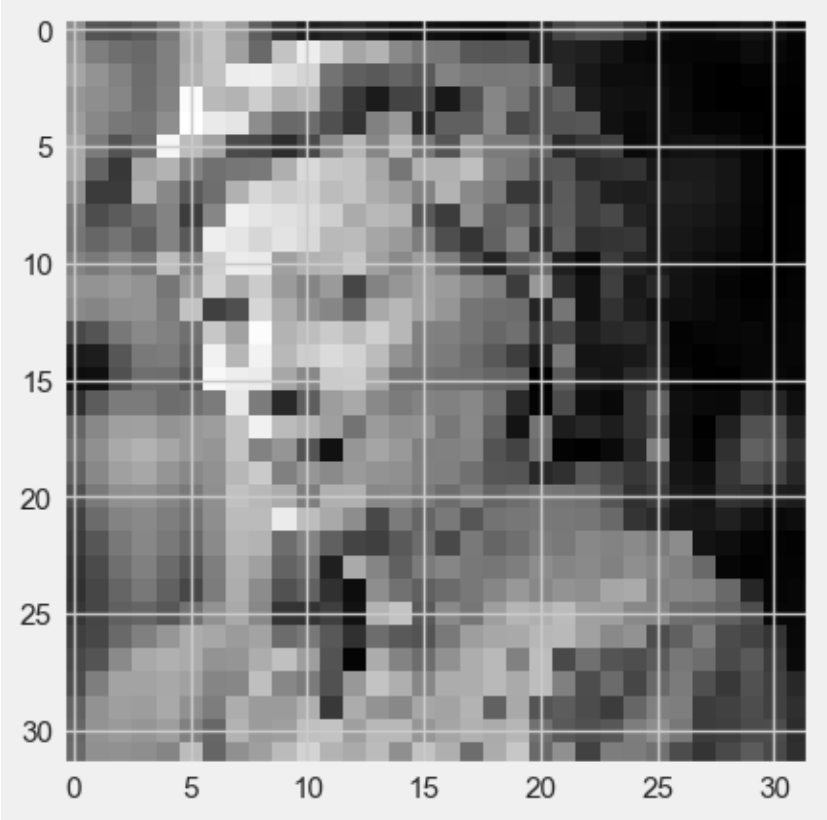
```
#display the first test image
plt.imshow(testimgs[0], cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x1d8806ee490>
```



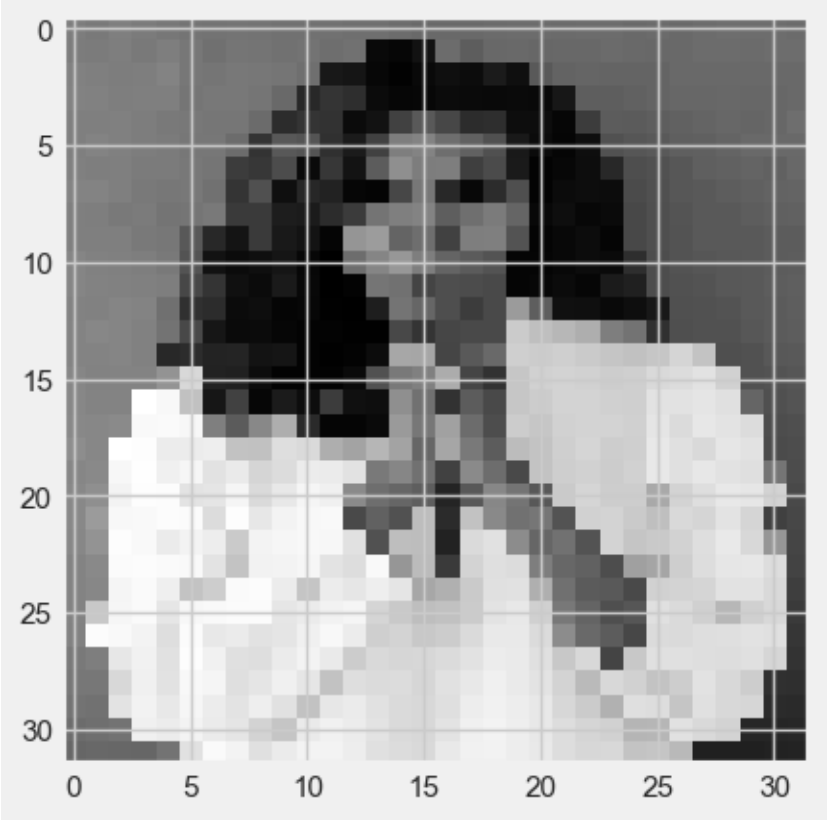

```
#display the second test image
plt.imshow(testimgs[1], cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x1d8806a9610>
```



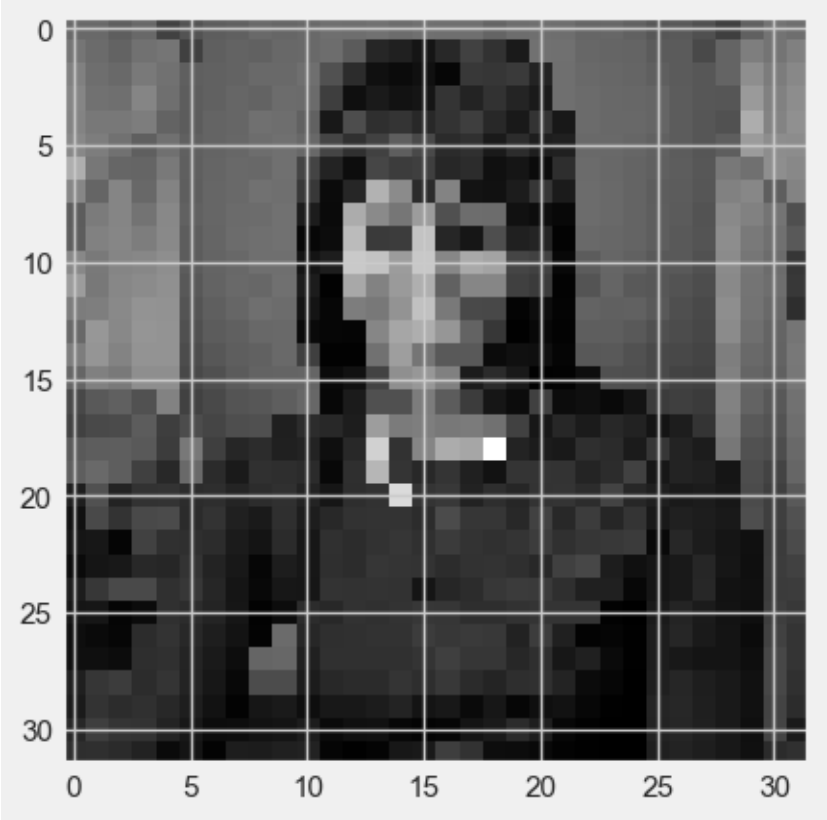

```
#display the third test image
plt.imshow(testimgs[2], cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x1d8fd121610>
```




```
#display the fourth test image
plt.imshow(testimgs[3], cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x1d8fd179610>
```




```
Xtest=[testdata.loc[:,"0"],testdata.loc[:,"1"],testdata.loc[:,"2"],testdata.loc[:,"3
 ↪"]]
model.predict(Xtest)
```

```
array([1, 0, 0, 1])
```

Note that in the output prediction, 0=Hillary Clinton and 1=Michelle Obama.

CHAPTER 11

DIFFERENTIAL EQUATIONS - Rachel Petrik

Note

Dr. Petrik (Rose-Hulman Institute of Technology) will cover the fundamentals of differential equations. A JNB lab was contributed by Jakob Hildebrandt.

11.1 Overview of Chapter

This Differential Equations Module offers a brief introduction to several important topics in undergraduate Differential Equations:

- Introduction to Differential Equations: Introduces the reader to the concept of differential equations and their applications, as well as basic terminology and classification. (Section 2)
- First-Order Differential Equations: Shows the reader how to solve first order differential equations, plot slope fields, and find equilibrium solutions (Section 3)
- Second-Order Differential Equations: Shows the reader how to solve second order differential equations (Section 4)
- Systems of Linear First Order Differential Equations: Shows the reader how to solve linear systems, find eigenvalues and eigenvectors, and draw and interpret phase planes and nullclines (Section 5)
- Laplace Transforms: Introduces the basic Laplace Transform, Inverse Laplace Transform, and Heaviside Function (Section 6)

For each topic, we provide basic definitions, examples, and exercises that are designed to be easily understood by entry level undergraduate students. The exercises include problems that are both solvable manually to check students' comprehension of the concepts and solvable using Python Jupyter Notebooks. The book ends with a lab that allows further practice of the topics taught in each chapter. This module focuses on the more pure mathematical aspects of Differential Equations, while the more applied side can be explored under the **Differential Equations for the Benefit of Society** Module in the **APPLICATIONS** chapter of this book.

11.2 Introduction to Differential Equations

```
import sympy as sym
```

Note

Differential equations form a fundamental branch of mathematics that deals with the modeling and analysis of change and variation. They play a crucial role in describing and understanding various phenomena across different fields, including physics, engineering, biology, economics, and many others. In simple terms, a differential equation is an equation that involves derivatives. It expresses a relationship between an unknown function and its derivatives, representing how the function changes with respect to one or more independent variables. These equations capture dynamic processes and allow us to study how quantities evolve and interact with each other.

Differential equations arise naturally when trying to model real-world problems. Applying mathematics to practical problems often leads to a situation where it's simpler to establish a connection between a function and its rate of change, rather than figuring out the function's exact formula. This relationship, between an unknown function and its derivatives, essentially gives rise to a differential equation.

Solving differential equations involves finding the function or functions that satisfy the equation. This task can be challenging as it often requires advanced mathematical techniques, including integration, differentiation, linear algebra, and complex analysis. Differential equations can exhibit a wide range of behaviors. Some equations have simple and explicit solutions, while others may require numerical methods or approximation techniques. Understanding the solutions to differential equations provides valuable insights into the underlying processes and helps make predictions about future behavior.

Moreover, differential equations serve as a bridge between theoretical mathematics and real-world applications. They enable us to formulate mathematical models that accurately represent complex systems and phenomena, allowing us to analyze, optimize, and make informed decisions. For instance, in physics, they describe the motion of objects, the flow of fluids, and the behavior of electromagnetic fields. In biology, differential equations help us analyze population growth, the spread of diseases, and the dynamics of biological systems. Engineers rely on differential equations to study control systems, electrical circuits, and structural stability.

11.2.1 Basic Terminology

A **differential equation** is any equation involving an unknown function and one or more of its derivatives. Differential equations can be largely classified into two categories: ordinary differential equations and partial differential equations. When the unknown function within the equation only relies on a single variable, we label it as an **ordinary differential equation**. However, when the unknown function depends on multiple independent variables, we refer to it as a **partial differential equation**. In this latter scenario, the derivatives present in the equation are identified as partial derivatives.

Example 1

Example 1

- $y' = x^3$ is an ordinary differential equation.
- $\frac{dx}{dt} = kx^2 \sin(t)$ is an ordinary differential equation.
- $y'(x) + \mu\sqrt{y(x)} = 0$ is an ordinary differential equation.
- $\frac{\partial Q}{\partial t} + Q(s,t) = 5$ is a partial differential equation.

Note that in this chapter we will primarily be working with ordinary differential equations. A differential equation can consist of three distinct types of quantities. First, the unknown function, which the equation aims to solve, is referred to as the **dependent variable**. In the context of ordinary differential equations, this dependent variable is a function of a single **independent variable**. Beyond the independent and dependent variables, there's a third variable known as a **parameter** that may feature in the equation. A parameter is a value that stays constant within a specific problem, but it can vary when comparing different problems. The **order** of a differential equation is determined by the rank of the highest derivative of the unknown function included in the equation.

Example 1 (continued)

Example 1 (continued)

- $y''(x) = x^3$ is an ODE. The dependent variable is y. The independent variable is x. There are no parameters and the order is 2.
- $\frac{dx}{dt} = kx^2 \sin(t)$ is an ODE. The dependent variable is x. The independent variable is t. There is one parameter, k. The order is 1.
- $y^{(5)}(x) + \mu\sqrt{y(x)} = 0$ is an ODE. The dependent variable is y. The independent variable is x. There is one parameter, μ. The order is 5.

An nth order ordinary differential equation is said to be **linear** if it can be written in the form

$a_n(x)y^{(n)}(x) + a_{n-1}(x)y^{(n-1)}(x) + ... + a_1(x)y'(x) + a_0(x)y(x) = g(x).$

If it cannot be written in that form, we say that the ordinary differential equation is **nonlinear**.

Example 1 (continued)

Example 1 (continued)

- $y''(x) = x^3$ is a linear ODE.
- $\frac{dx}{dt} = kx^2 \sin(t)$ is a nonlinear ODE. This is because x, the dependent variable, is raised to the second power.
- $y^{(5)}(x) + \mu\sqrt{y(x)} = 0$ is a nonlinear ODE. This is because y is under a radical sign. Note that the parameters in either problem, k and μ, do not affect the linearity of either of these last two problems.

A **solution** to a differential equation is a sufficiently differentiable function that, if substituted into the equation together with its necessary derivatives, makes the equation an identity over some interval of the independent variable.

Example 2

Example 2

Show that $f(x) = e^{-x}$ is a solution for all x to $2y' + 3y = e^{-x}$.

```
This is the function f(x): exp(-x)
This is the derivative of f(x): -exp(-x)
```

```
True
```

11.2.2 Exercises

Exercises

1. List the type and order of each of the following differential equations. For each ordinary differential equation, determine if it is linear or nonlinear.
   - $\frac{dQ}{dt} = \frac{tP}{100}$
   - $\frac{\partial^2 v}{\partial t^2} + \frac{\partial^2 v}{\partial w^2} = 5$
   - $2\frac{d^2q}{dt^2} + \cos(t)\frac{dq}{dt} = E(t)$
2. Show that $f(x) = \sin(x) + \cos(x) + x^2 - 2$ is a solution for all x to $y'' + y = x^2$.

11.3 First-Order Differential Equations

Note

In this chapter, methods will be given for solving first-order differential equations. Remember that first-order means that the first derivative of the unknown function is the highest derivative appearing in the equation.

11.3.1 Separable First-Order Differential Equations

A first order differential equation is called **separable** if it can be written as $\frac{dx}{dt} = f(x) \cdot g(t)$, that is, the independent variable and the dependent variable of the differential equation can be *separated* from one another.

Example

- $y' = x^3$ is a separable differential equation.
- $\frac{dx}{dt} = kx^2 \sin(t)$ is a separable differential equation.
- $y'(x) = x^2 - 3y$ is not a separable differential equation.
- $\frac{dx}{dt} \cdot t - 3xt^2 = 7t^2$ is a separable differential equation. In particular, this can be rewritten as $\frac{dx}{dt} = (7 + 3x)t$.

Given a separable differential equation $\frac{dx}{dt} = f(x) \cdot g(t)$, we can solve it with the following procedure.

1. Separate the variables: $\frac{1}{f(x)}dx = g(t)dt$

   Note: $\frac{dx}{dt}$ is not a fraction, but this process can be done because we are implicitly applying the chain rule for multivariate functions.

2. Integrate both sides with respect to the appropriate variable: $\int \frac{1}{f(x)}dx = \int g(t)dt.$

3. Write the solution in the form $F(x) = G(t) + C$, where $F'(x) = \frac{1}{f(x)}$ and $G'(t) = g(t)$.
4. If possible, solve the equation explicitly for the dependent variable, in this case x.

Example

Find the explicit general solution to the differential equation $\frac{dx}{dt} = xt^2$.

Separate: $\frac{1}{x}dx = t^2dt$

Integrate: $\int \frac{1}{x}dx = \int t^2dt$

$\ln|x| = \frac{1}{3}t^3 + C \rightarrow$ *Note: This is the implicit solution to the differential equation.*

Solve for the Dependent Variable:

$$\ln|x| = \frac{1}{3}t^3 + C$$

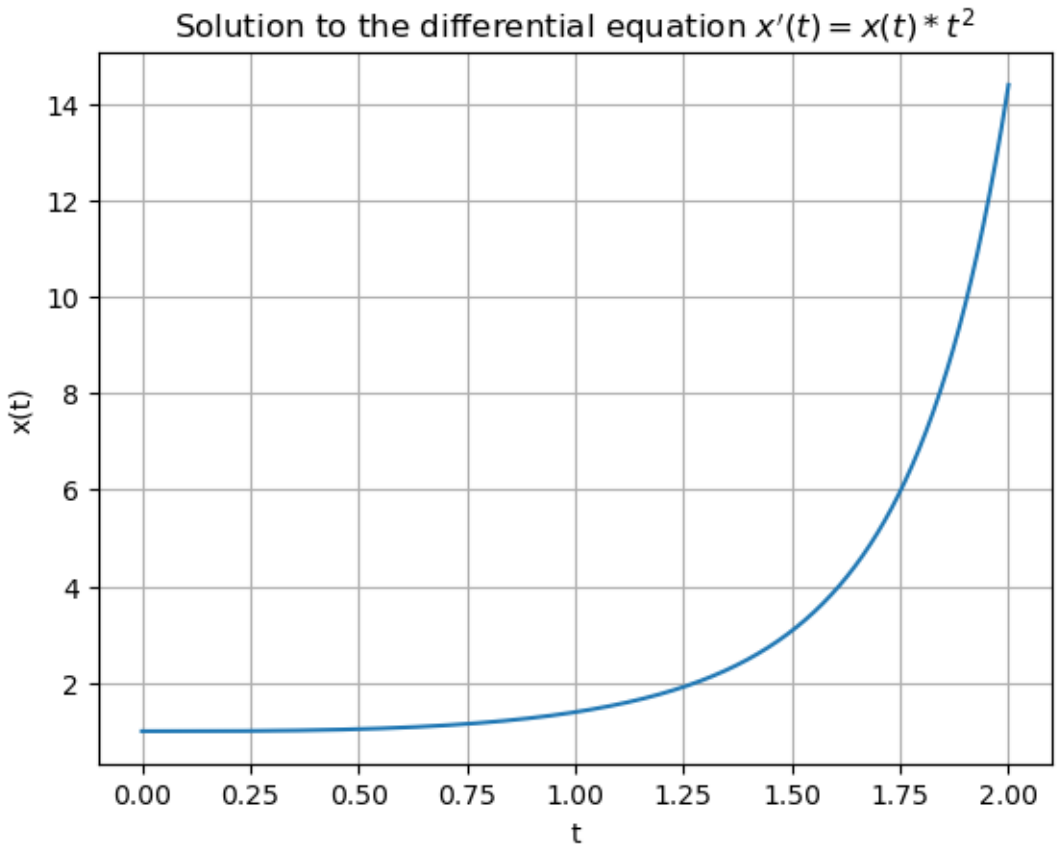
$$e^{\ln|x|} = e^{\frac{1}{3}t^3 + C}$$

$$|x| = e^{\frac{1}{3}t^3}e^C$$

$$x = \pm e^C e^{\frac{1}{3}t^3}$$

$$x = De^{\frac{1}{3}t^3}, \text{ where } D = \pm e^C$$

```
Eq(x(t), C1*exp(t**3/3))
```



Example

Find the explicit solution to the initial value problem $\frac{dW}{ds} = \frac{2s\cos^2(W)}{s^2-1}$ where $W(0) = 0$.

Separate: $\frac{1}{\cos^2(W)}dW = \frac{2s}{s^2-1}ds$

Integrate: $\int \sec^2(W)dW = \int \frac{2s}{s^2-1}ds$

$\tan(W) = \ln|s^2 - 1| + C \rightarrow$ *Note: This is the implicit solution to the differential equation.*

Solve for the Dependent Variable:

$$W = \arctan(\ln|s^2 - 1| + C)$$

Use Initial Condition $W(0) = 0$:

$$0 = \arctan(\ln|-1| + C)$$

$$0 = \arctan(C)$$

$$0 = C$$

Explicit Solution: $W = \arctan(\ln|s^2 - 1|)$

11.3.2 Linear First-Order Differential Equations

A first order differential equation is called **linear** if it can be written as $a_1(t)\frac{dx}{dt} + a_0(t)x = g(t)$. The equation is linear in the dependent variable, in this case x, and its derivative; the coefficient functions $a_1(t)$, $a_0(t)$, and $g(t)$ can be any function involving t.

Example

- $y' + ye^x = \tan(t)$ is a linear differential equation.
- $x\frac{dx}{dt} = kx^2\sin(t)$ is not a linear differential equation because of the $x\frac{dx}{dt}$ term.
- $x'(t) = x^2 - 3t$ is not a linear differential equation because of the x^2 term.
- $\frac{dx}{dt}\cdot t - 3xt^2 = 7t^2$ is a linear differential equation. In particular, this can be rewritten as $t\frac{dx}{dt} - 3t^2x = 7t^2$.

A first-order linear differential equation is said to be in **standard form** if it is written in the form $\frac{dx}{dt} + p(t)x = q(t)$. In other words, if the coefficient on the first derivative is 1. Note that any first-order linear differential equation can be written in standard form by performing simple algebra.

If $q(t) = 0$, we say that the differential equation is **homogeneous**. If $q(t) \neq 0$, we say that the differential equation is **nonhomogeneous**. Note that if a first order differential equation is both linear and homogeneous, then it is also separable. You will be asked to show *why* in the exercises.

We will now develop a strategy for solving first-order linear differential equations. The basic idea is that we will multiply by a positive function that simplifies the left hand side of the differential equation. We must choose a positive function in order to preserve the solutions to the differential equation.

Given a first-order linear differential equation *in standard form* $x' + p(t)x = q(t)$, the **integrating factor** for the differential equation is the function $\mu(t) = e^{P(t)}$ where $P(t)$ is *any* antiderivative of $p(t)$, that is, $P'(t) = p(t)$. The integrating factor is the positive function that we will multiply our differential equation by. So what is the significance of multiplying by the integrating factor?

Consider $x' + p(t)x = q(t)$.

1. Multiply by $\mu(t) = e^{P(t)}$, which yields $\mu(t)x' + \mu(t)p(t)x = \mu(t)q(t)$
2. Observe $\mu'(t) = e^{P(t)}P'(t) = e^{P(t)}p(t) = \mu(t)p(t)$.
3. Rewrite $\mu(t)x' + \mu(t)p(t)x = \mu(t)q(t)$ as $\mu(t)x' + \mu'(t)x = \mu(t)q(t)$.
4. Observe $\frac{d}{dt}\left(\mu(t)x(t)\right) = \mu(t)x'(t) + \mu'(t)x(t)$.
5. Rewrite $\mu(t)x' + \mu'(t)x = \mu(t)q(t)$ as $\frac{d}{dt}\left(\mu(t)x(t)\right) = \mu(t)q(t)$.
6. Integrate $\displaystyle\int \frac{d}{dt}\left(\mu(t)x(t)\right)dt = \int \mu(t)q(t)dt$.

$$\mu(t)x(t) = \int \mu(t)q(t)dt$$

7. Solve for the dependent variable

$$x(t) = \frac{1}{\mu(t)}\int \mu(t)q(t)dt.$$

Note we have no guarantee that we can integrate $\int \mu(t)q(t)dt$, but if we can we are able to solve first-order linear differential equations.

Example

Find the general solution to the differential equation $\frac{dx}{dt} + 3x = te^{-3t}$.

Note: This differential equation is in standard form so we can proceed by finding the integrating factor.

1. Multiply by $\mu(t) = e^{\int 3dt} = e^{3t}$.

$$e^{3t}x'(t) + 3e^{3t}x(t) = t$$

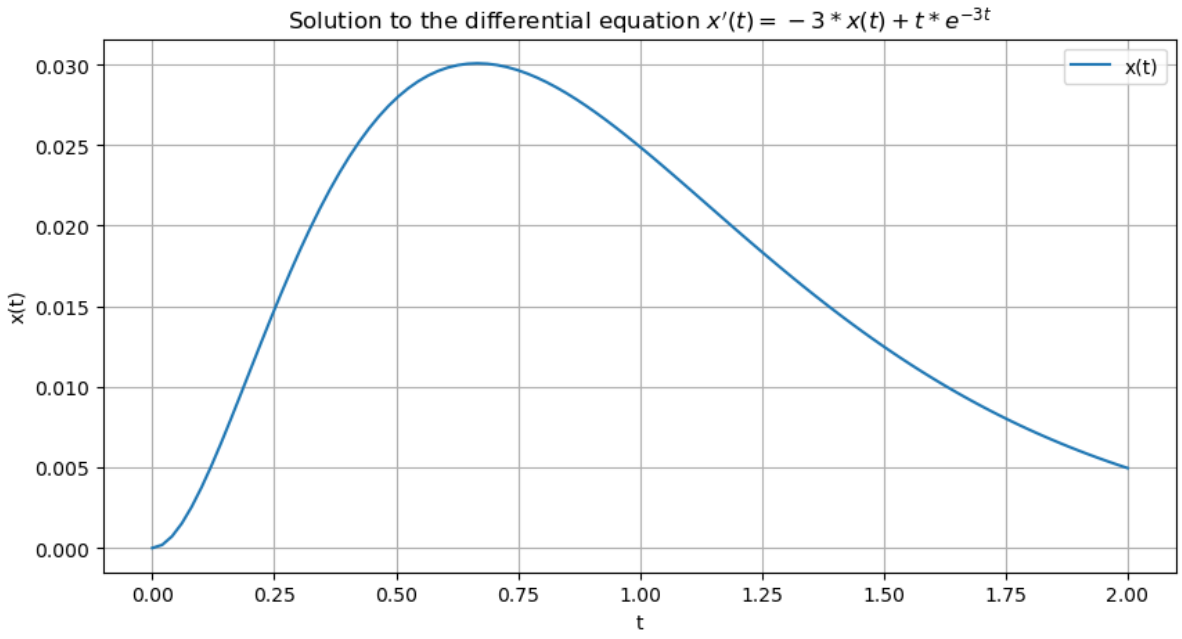
2. Observe $\frac{d}{dt}(e^{3t} \cdot x(t)) = e^{3t}x'(t) + 3e^{3t}x(t)$.
3. Rewrite $e^{3t}x'(t) + 3e^{3t}x(t) = t$ as $\frac{d}{dt}(e^{3t} \cdot x(t)) = t$.
4. Integrate $\int \frac{d}{dt}(e^{3t} \cdot x(t))dt = \int tdt$.

$$e^{3t}x(t) = \frac{1}{2}t^2 + C$$

5. Solve for the dependent variable.

$$x(t) = e^{-3t}(\frac{1}{2}t^2 + C)$$

```
Eq(x(t), (C1 + t**2/2)*exp(-3*t))
```



11.3.3 Slope Fields

Note we still do not know how to solve every first-order linear differential equation. For example, $y' = x(x^2 - y^2)$ is neither separable or linear. However, even if we cannot solve a differential equation, it is still possible to get qualitative information about the solution.

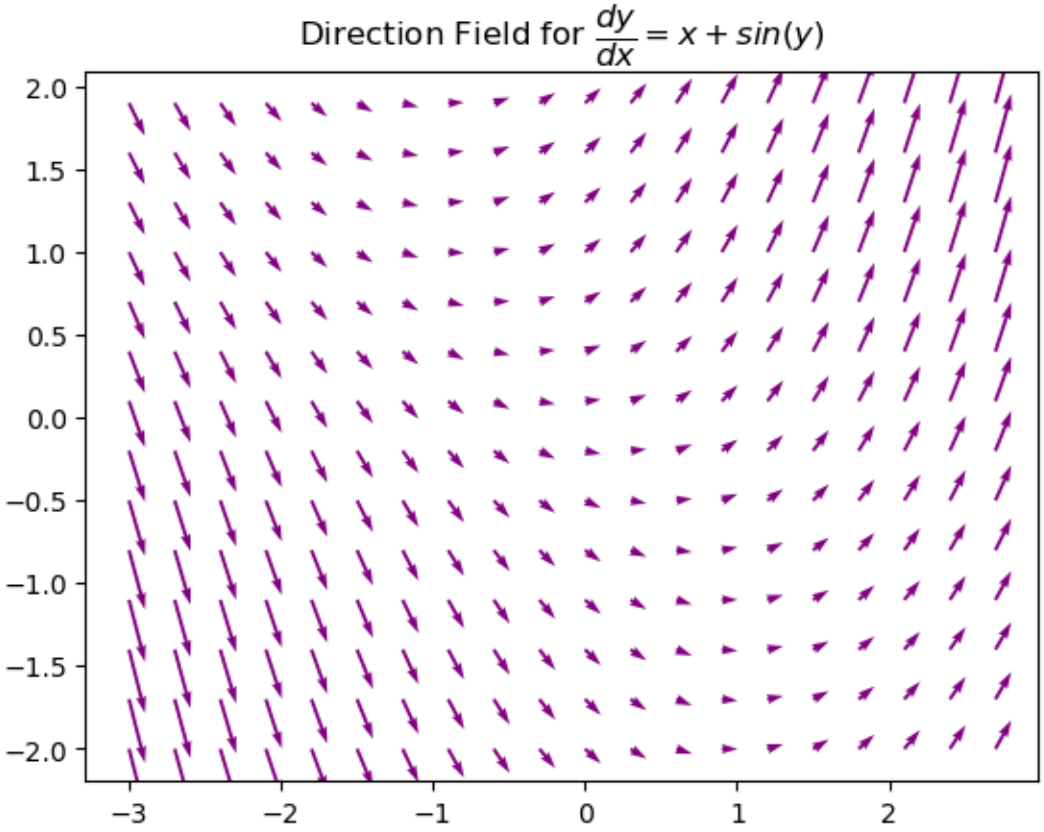
Consider the first-order differential equation $\frac{dx}{dt} = f(t, x)$. If we evaluate $f(t, x)$ at a set of grid points, we can then draw short vectors that represent the slope of the solution $x(t)$ at those points. If we look at this grid of vectors at many points, we can get an idea of what the solution, $x(t)$, looks like. This grid of vectors is called a **slope field** or **direction field**.

To draw a direction field, you can simply evaluate the differential equation at various points and plot the resulting vectors. For example, using the equation $y' = x(x^2 - y^2)$, if you plug in the point (0,0), you see that the slope (y') is 0 at that point, so you can draw a short vector of 0 slope at the origin. In fact, whenever $x = 0$, the slope is 0, so along the y-axis, there will be short horizontal (0 slope) vectors at every y-value. This process can be repeated for a set of sample points to create a graph that tells us about the behavior of solutions to a first order equation.

Direction fields can also be plotted using electronically, as demonstrated below.

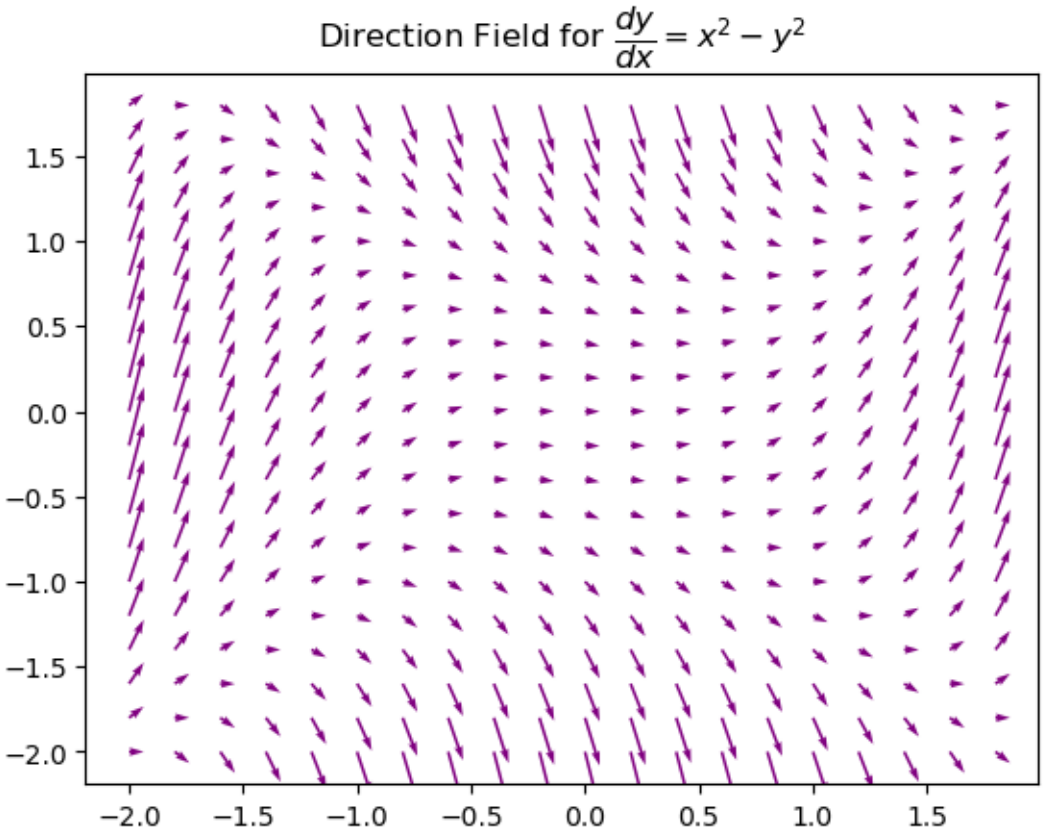
Example

Plot a direction field for $y'(x) = x + \sin(y)$.



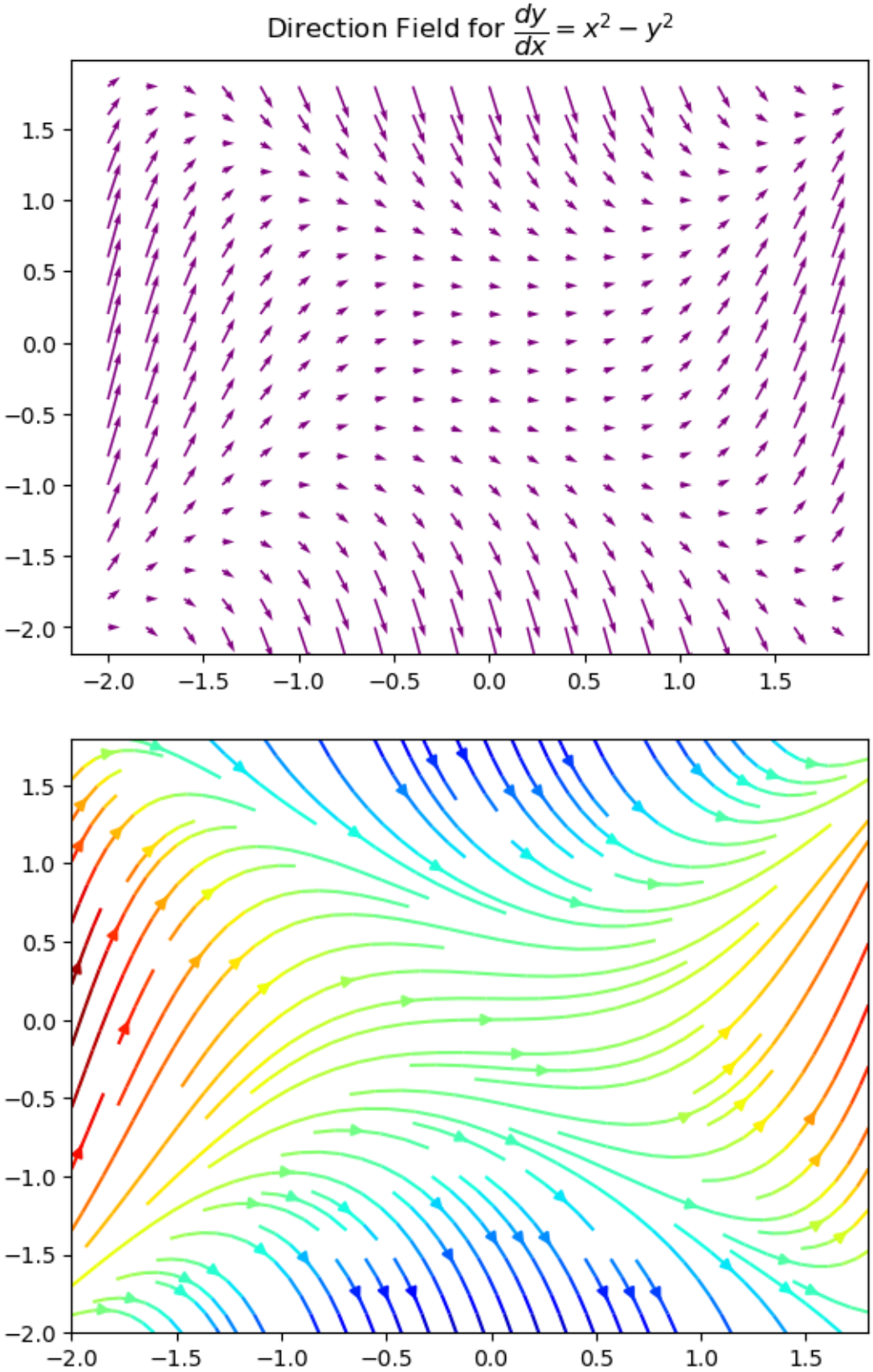
Example

Plot a direction field for $y'(x) = x^2 - y^2$.



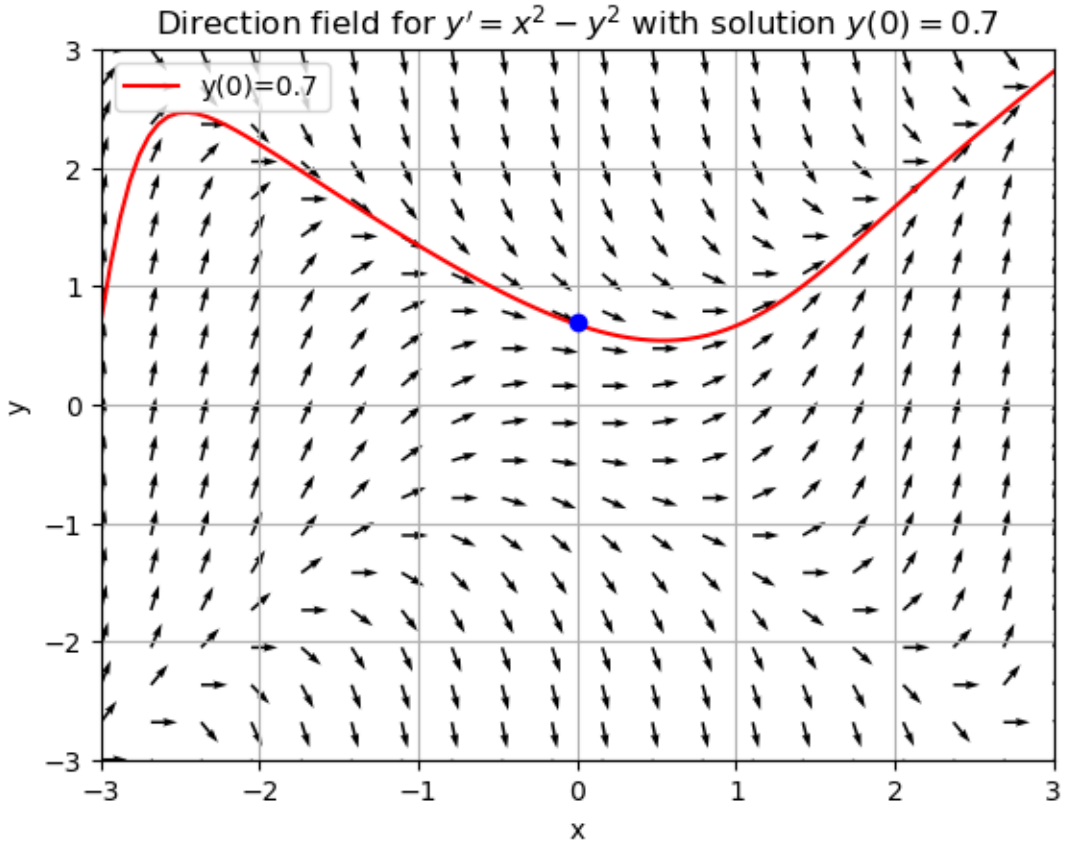
Example

Plot a direction field for $y'(x) = x^2 - y^2$.



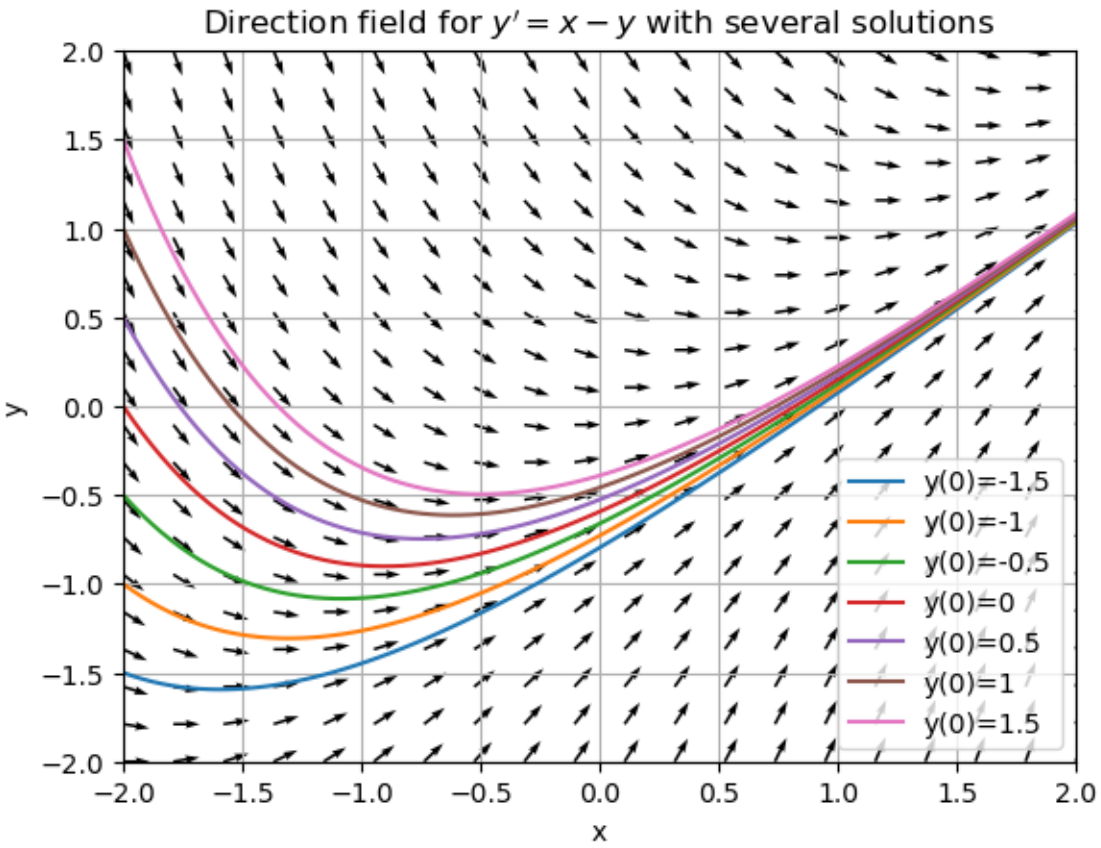
You may use the following code to plot a direction field for $y'(x) = x^2 - y^2$ and a particular solution curve given an

initial condition.



Example

Plot a direction field for $y'(x) = x - y$ with multiple solution curves.



You may also use the following code to plot a direction field for $y'(x) = x - y$ and a single solution curve. Notice that there is a slider which allows you to adjust the initial condition and see different solution curves based on your choice of initial condition.

Note

If you are reading this textbook in a format that does not allow this code cell to be executed (such as a webpage or a PDF), please download the original Jupyter Notebook or launch it in Binder to engage with this interactive resource.

```
interactive(children=(FloatSlider(value=0.0, description='y0', max=2.0, min=-2.0),␣
↪Output()), _dom_classes=('w…
```

11.3.4 Autonomous Equations

A first-order differential equation of the form $\frac{dx}{dt} = f(x)$ is called **autonomous**, that is, a differential equation where only the dependent variable appears. We can always solve these because they are always separable. In particular, we can always rewrite the differential equation as $\int \frac{1}{f(x)} dx = \int dt$ assuming $f(x) \neq 0$. But it can sometimes be very difficult to do so.

It turns out that autonomous differential equations are nice in that they have constant solutions. In particular, $x(t) = c$ is a solution if $x'(t) = \frac{dc}{dt} = 0 = f(x(t)) = f(c)$. These are the trivial solutions we would lose when dividing by zero when handling it as a separable differential equation.

An **equilibrium solution** of an autonomous differential equation is a constant solution to the differential equation. In other words, if $f(c) = 0$, then $x(t) = c$ is an equilibrium solution of $x'(t) = f(x)$.

Example

Find the equilibrium solutions of $x'(t) = x^2 - 9$.

$x'(t) = 0$

$x^2 - 9 = (x-3)(x+3) = 0$

So $x(t) = -3$ and $x(t) = 3$ are the equilibrium solutions.

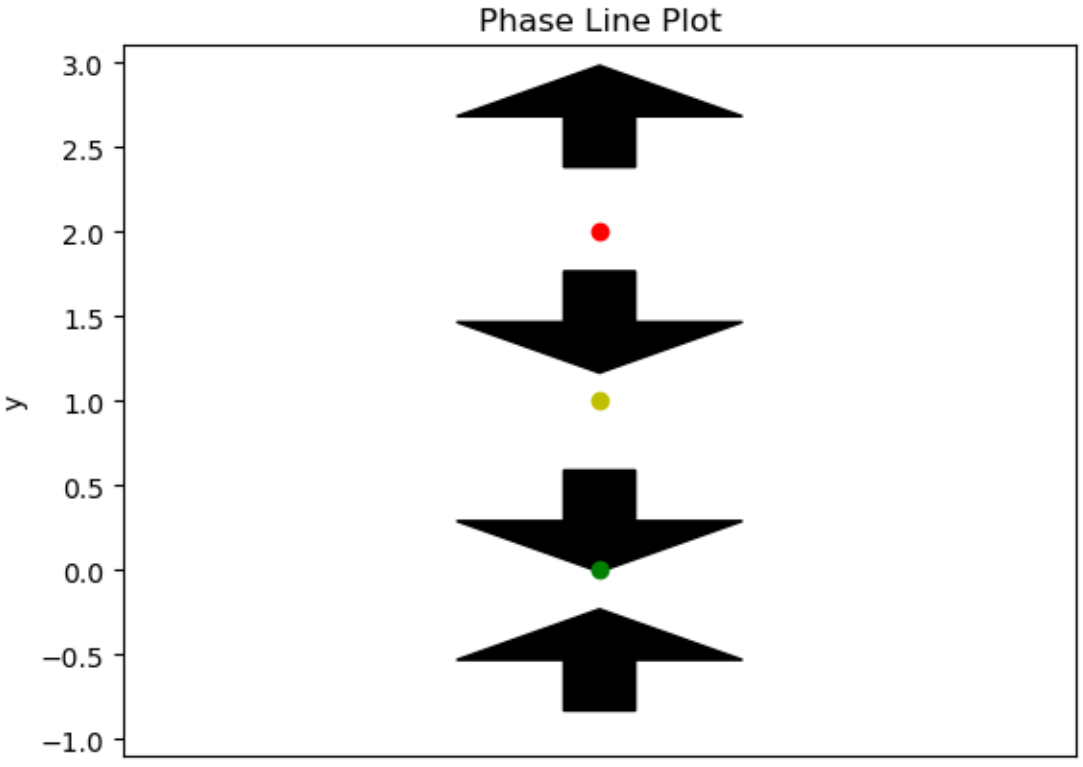
*Note: $\int \frac{1}{x^2-9} dx = \int dt$ would not have recovered these solutions.

Given the equilibrium solutions of an autonomous differential equation, we can make a phase line to analyze the behavior of the solution.

Example

Draw a phase line and determine where the solutions of the differential equation $y'(x) = y(y-1)^2(y-2)$ are increasing and decreasing.

First we must find the equilibrium solutions, which means we must solve $y'(x) = 0$. This means $y(y-1)^2(y-2) = 0$. So $y(x) = 0$, $y(x) = 1$, and $y(x) = 2$ are the equilibrium solutions. To draw a phase line, see the code below.



Notice that this tells us that a solution to the previous differential equation would have to be increasing on the interval $(-\infty, 0)$ or decreasing on $(0, 1)$ or decreasing on $(1, 2)$ or increasing on $(2, \infty)$.

In addition it allows us to classify the stability of our equilibrium solutions. An equilibrium solution $x(t) = c$ is **stable** if solutions whose initial value is near c stays near c as the independent variable increases. An equilibrium solution $x(t) = c$ is **unstable** if solutions whose initial value is near c goes away from c as the independent variable increases. An equilibrium solution is considered **semi-stable** if both cases occur (one in the region above and one in the region below). We can see an example of each type of equilibrium in the above example. In particular, $x(t) = 0$ is stable, $x(t) = 1$ is semi-stable, and $x(t) = 2$ is unstable. Alternatively, a stable solution can be referred to as a **sink**, an unstable solution as a **source**, and a semi-stable solution as a **node**.

It turns out that if we know that there is a unique solution to the autonomous initial-value problem $\frac{dx}{dt} = f(x)$ where $x(t_0) = x_0$, then we know a few more things about the solution. Namely,

1. The equilibrium points divide the region $\mathbb{R}^2$ plane into regions. A solution cannot pass between regions and the initial value will determine which region a particular solution is in.
2. The solution $x(t)$ is increasing/decreasing everywhere inside a particular region because $f(x)$ is either positive or negative everywhere in a region.
3. The limit of a solution $x(t)$ as $t \to \pm\infty$ is one of the boundaries of that region.

11.3.5 Exercises

Exercises

1. Determine if the following first-order differential equations are linear, separable, both, or neither.
   - $\frac{dy}{dx} = y + x$
   - $\sqrt{y} + 2\frac{dy}{dx} = e^x$
   - $\sqrt{y}\frac{dy}{dx} = e^x$
   - $\frac{dy}{dx} = x + 5$
2. Find the explicit solution to the differential equation $y'(x) = y + 5$.
3. Find the explicit solution to the differential equation $W'(s) = s + sW^2$.
4. Find the explicit solution to the initial value problem $y'(t) = ty$ where $y(0) = 3$.
5. Find the explicit solution to the initial value problem $x'(t) = x\sin(t)$ where $x(0) = 1$.
6. Show that if a first order differential equation is both linear and homogeneous, then it is also separable.
7. Find the solution to the differential equation $\frac{dP}{dt} = \frac{2P}{t} + t^2\sec^2(t)$.
8. Find the solution to the initial value problem $xy' + y = \sin(x)$, where $y(\pi) = 0$.
9. Plot a direction field for $y'(x) = y + x$.
10. Plot a direction field for $y'(x) = xy$.
11. Find the equilibrium solutions of $y'(x) = \sin(y)$.
12. Draw a phase line and determine where a solution of the differential equation $\frac{dT}{dt} = (T + 1)T(T - 3)^2$ is increasing and decreasing. Then classify the stability of each equilibrium point.

11.4 Second-Order Differential Equations

Note

In this section, we will start with some general theory of nth-order linear differential equations to motivate the techniques used for solving second-order differential equations. Second-order differential equations play a critical role in mathematical physics and engineering disciplines due to their ubiquity in modeling natural phenomena. They are differential equations in which the highest order derivative is of the second order. Specifically, a second-order differential equation can be written as $\frac{d^2x}{dt^2} = f(t, x, \frac{dx}{dt})$ where x is the unknown function we are interested in, t is the independent variable, $\frac{dx}{dt}$ is the first derivative of x, and $\frac{d^2x}{dt^2}$ is the second derivative of x.

Examples of the practical applications of second-order differential equations include but are not limited to: modeling the motion of a spring (harmonic oscillator), predicting the population dynamics in ecology, and describing the behavior of electric circuits. They are also widely used in the fields of physics, economics, and engineering to create mathematical models of real-world situations. One reason for the importance of second-order equations is that if $x(t)$ represents the position of an object at time t, then $x'(t)$ is its instantaneous velocity and $x''(t)$ is its acceleration; thus, real-world problems from the physical sciences often lead to equations of this type.

Solving second-order differential equations can be quite complex and may involve techniques such as characteristic equations, method of undetermined coefficients, and variation of parameters, among others. The nature of the solution can widely vary depending on the form of the equation. For instance, solutions can be real or complex, periodic or exponential, and convergent or divergent. Some special types of second-order differential equations, like the Cauchy-Euler equation or the Bessel's equation, have their own dedicated methods of solutions due to their unique structure.

As we explore the world of second-order differential equations, we uncover a variety of mathematical landscapes with significant implications for understanding the world around us. Through these equations, we can dive deep into the patterns and behaviors that underscore numerous phenomena, both natural and man-made.

Recall that a general linear nth-order initial-value problem is of the form

$$a_n(x)y^{(n)}(x) + a_{n-1}y^{(n-1)}(x) + \cdots + a_1(x)y'(x) + a_0(x)y(x) = g(x)$$

where $y(x_0) = y_0$, $y'(x_0) = y_1$, ..., $y(n-1)(x_0) = y_{n-1}$. Recall that the number of initial conditions must match the order of the differential equation and they must have the same input x_0. We call a linear differential equation **homogeneous** if $g(x) = 0$, or in other words, if it can be written as

$$a_n(x)y^{(n)}(x) + a_{n-1}y^{(n-1)}(x) + \cdots + a_1(x)y'(x) + a_0(x)y(x) = 0.$$

11.4.1 General Theory

One of the most important principles for homogeneous equations is the **Superposition Principle** which states that if $y_1, y_2 \ldots, y_n$ are solutions to the homogeneous linear nth-order differential equation $a_n(x)y^{(n)}(x) + a_{n-1}y^{(n-1)}(x) + \cdots + a_1(x)y'(x) + a_0(x)y(x) = 0$ on an interval I. Then the linear combination $y(x) = c_1y_1(x) + c_2y_2(x) + \cdots + c_ny_n(x)$ where c_k is an arbitrary constant for each k is also a solution on I. In other words, we can build new solutions from existing solutions. This idea is critical in solving second-order differential equations.

Some nice consequences of the Superposition Principle are

1. If y is a solution to a homogeneous differential equation, then so is cy for any constant c. In other words, we can scale every solution to produce another solution.
2. $y = 0$ is a solution to every homogeneous differential equation. This comes from setting $c_k = 0$ for all k.

Example

Consider the differential equation $y'' - 4y = 0$. Verify that $y(x) = e^{2x}$ and $y(x) = e^{-2x}$. Use the Superposition Principle to conclude that $y(x) = Ae^{2x} + Be^{-2x}$ is also a solution for any constants A and B.

```
This is the function f(x): exp(2*x)
This is the derivative of f(x): 2*exp(2*x)
This is the second derivative of f(x): 4*exp(2*x)
```

```
True
```

```
This is the function f(x): exp(-2*x)
This is the derivative of f(x): -2*exp(-2*x)
This is the second derivative of f(x): 4*exp(-2*x)
```

```
True
```

The Superposition Principle tells us that we can build new solutions by taking linear combinations of known solutions. This means that $y(x) = Ae^{2x} + Be^{-2x}$ is a solution since $y(x) = e^{2x}$ and $y(x) = e^{-2x}$ are known solutions.

11.4.2 Linear Homogeneous Differential Equations with Constant Coefficients

We expect solutions of an nth-order linear homogeneous differential equation with constant coefficients $a_n x^{(n)} + a_{n-1}x^{(n-1)} + \cdots + a_1 x' + a_0 x = 0$ to be of the form $x(t) = e^{kt}$ for some constant k. We also know that if $x_1(t) = e^{k_1 t}$ and $x_2(t) = e^{k_2 t}$ are both solutions, then $x(t) = c_1 e^{k_1 t} + c_2 e^{k_2 t}$, where c_1 and c_2 are any constants, is also a solution by the Superposition Principle.

Motivating Example

Find all solutions to $x'' - 7x' + 12x = 0$.

Suppose $x(t) = e^{kt}$ is a solution. This means it satisfies our differential equation. Since $x'(t) = ke^{kt}$ and $x''(t) = k^2 e^{kt}$, we know

$x'' - 7x' + 12x = k^2 e^{kt} - 7ke^{kt} + 12e^{kt} = e^{kt}(k^2 - 7k + 12) = 0.$

Since $e^{kt} \neq 0$, we know that $k^2 - 7k + 12 = 0$ when e^{kt} is a solution. We call the equation $k^2 - 7k + 12 = 0$ the **characteristic equation** of the differential equation. Solving the characteristic equation gives $k^2 - 7k + 12 = (k-3)(k-4) = 0$. Thus, $k = 3$ or $k = 4$ meaning that $x(t) = e^{3t}$ and $x(t) = e^{4t}$ are both solutions to the differential equation. Moreover, by the Superposition Principle, $x(t) = c_1 e^{3t} + c_2 e^{4t}$ are also solutions for all constants c_1 and c_2.

So a natural question is did we find all solutions? We found an infinite family of solutions, but how do we know there are not any other solutions? We will be exploring this question next.

A set of functions is said to be **linearly dependent** if one function can be written as a linear combination of the others. This means that a set of function $\{f_1(t), f_2(t), \ldots, f_n(t)\}$ is linearly dependent if there exist constants $c_1, c_2, \ldots, c_n$ **not all zero** such that $c_1 f_1(t) + c_2 f_2(t) + \cdots + c_n f_n(t) = 0$. A set of functions is said to be **linearly independent** if the only way $c_1 f_1(t) + c_2 f_2(t) + \cdots + c_n f_n(t) = 0$ is true is if all of the c_i are zero.

Example

- $\{1, t, 4t+2\}$ is a linearly dependent set of functions because 2(1)+4(t)-(4t+2)=0.
- $\{1, x, x^2\}$ is a linearly independent set of functions because $c_1 + c_2 x + c_3 x^2 = 0$ only when $c_1 = c_2 = c_3 = 0$.

It turns out that this idea of a linearly dependent/independent set of functions allows us to precisely answer the earlier question of "Did we find all of the solutions?" through the following theorem.

Theorem

Every nth-order linear homogeneous equation has a set of n linearly independent solutions; this is called a **fundamental set of solutions**. Let $x_1, x_2, \ldots, x_n$ be a fundamental set of solutions of an nth-order linear homogeneous equation. By the Superposition Principle, $x(t) = c_1 x_1(t) + c_2 x_2(t) + \cdots + c_n x_n(t)$ is also a solutions; we call this the **general solution**.

Let's now return to our motivating example.

Motivating Example (continued)

Find all solutions to $x'' - 7x' + 12x = 0$.

Recall previously we found that $x(t) = e^{3t}$ and $x(t) = e^{4t}$ are both solutions to the differential equation. Moreover, by the Superposition Principle, $x(t) = c_1 e^{3t} + c_2 e^{4t}$ are also solutions for all constants c_1 and c_2.

Since this differential equation is second-order, we know that the fundamental set of solutions contains two linearly independent solutions. Notice that e^{3t} and e^{4t} are linearly independent as $c_1 e^{3t} + c_2 e^{4t} = 0$ only when $c_1 = c_2 = 0$. This means that $\{e^{3t}, e^{4t}\}$ *is* the fundamental set of solutions and thus, the general solution $x(t) = c_1 e^{3t} + c_2 e^{4t}$ does capture *all* solutions of this differential equation.

We will now look at two other examples that will ultimately motivate our strategy for these second-order differential equations.

Example

Find all solutions to $x'' - 6x' + 9x = 0$.

Suppose $x(t) = e^{kt}$ is a solution. This means it satisfies our differential equation. Since $x'(t) = ke^{kt}$ and $x''(t) = k^2 e^{kt}$, we know

$$x'' - 6x' + 9x = k^2 e^{kt} - 6ke^{kt} + 9e^{kt} = e^{kt}(k^2 - 6k + 9) = 0.$$

Since $e^{kt} \neq 0$, we know that $k^2 - 6k + 9 = 0$ when e^{kt} is a solution. Thus, $k^2 - 6k + 9 = 0$ is the **characteristic equation** of this differential equation. Solving the characteristic equation gives $k^2 - 6k + 9 = (k-3)(k-3) = 0$. Thus, $k = 3$ meaning that $x(t) = e^{3t}$ is a solution to the differential equation. But, since this differential equation is second-order, we know that the fundamental set of solutions contains two linearly independent solutions. This means we are missing one.

We will now guess a second solution that is linearly independent from the first solution. Our guess is $y(t) = te^{3t}$. The motivation for the guess is that we expect the solution to look "like" $x(t) = e^{3t}$ and be linearly independent from it. But now we need to verify that our guess is actually a solution. If $y(t) = te^{3t}$ is a solution, then it satisfies our differential equation. Since $y'(t) = 3te^{3t} + e^{3t}$ and $y''(t) = 9te^{3t} + 6e^{3t}$, we know $x'' - 6x' + 9x = (9te^{3t} + 6e^{3t}) - 6(3te^{3t} + e^{3t}) + 9(te^{3t}) = 0$. Thus, $y(t) = te^{3t}$ is a solution.

This means that $\{e^{3t}, te^{3t}\}$ *is* the fundamental set of solutions and thus, the general solution $x(t) = c_1e^{3t} + c_2te^{3t}$ does capture *all* solutions of this differential equation.

Example

Find the *real-valued* general solution to $x'' + 9x = 0$.

Suppose $x(t) = e^{kt}$ is a solution. This means it satisfies our differential equation. Since $x'(t) = ke^{kt}$ and $x''(t) = k^2e^{kt}$, we know

$x'' + 9x = k^2e^{kt} + 9e^{kt} = e^{kt}(k^2 + 9) = 0.$

Since $e^{kt} \neq 0$, we know that $k^2 + 9 = 0$ when e^{kt} is a solution. Thus, $k^2 + 9 = 0$ is the **characteristic equation** of this differential equation. Solving the characteristic equation gives $k^2 + 9 = 0$. Thus, $k = \pm 3i$ meaning that $x_1(t) = e^{3it}$ and $x_2(t) = e^{-3it}$ are solutions to the differential equation. But, these are not *real-valued* solutions.

Recall Euler's formula which says that $e^{i\theta} = \cos(\theta) + i\sin(\theta)$. Using Euler's formula we can rewrite $x_1(t) = e^{3it} = \cos(3t) + i\sin(3t)$ and $x_2(t) = e^{-3it} = \cos(-3t) + i\sin(-3t)$. Moreover, since cosine is an even function, we know $\cos(-\theta) = \cos(\theta)$ and since sine is an odd function, we know $\sin(-\theta) = -\sin(\theta)$. Thus, $x_2(t) = \cos(3t) - i\sin(3t)$.

Recall that any linear combination of x_1, x_2 is also a solution by the superposition principle. This means that $y_1(t) = \frac{x_1(t)+x_2(t)}{2} = \cos(3t)$ is a solution *and* $y_2(t) = \frac{x_1(t)-x_2(t)}{2i} = \sin(3t)$ is also a solution. Note that $y_1(t)$ and $y_2(t)$ are still linearly independent and they are real-valued.

This means that $\{\cos(3t), \sin(3t)\}$ *is* the fundamental set of solutions and thus, the general solution $x(t) = c_1\cos(3t) + c_2\sin(3t)$ does capture *all* solutions of this differential equation.

Summary

Consider the differential equation $ax'' + bx' + cx = 0$.

1. The characteristic equation is $ak^2 + bk + c = 0$.
2. The possible solutions to the characteristic equation are $k = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$. There are three cases for what the roots of the characteristic equation can look like.
   - Case 1: $b^2 - 4ac > 0$. In this case the characteristic equation, has two distinct real roots.
   - Case 2: $b^2 - 4ac = 0$. In this case the characteristic equation, has one real root with multiplicity two.
   - Case 3: $b^2 - 4ac < 0$. In this case the characteristic equation, has two complex roots.
3. Based on 2, we can write down the possible forms of solutions to the differential equation.
   - Case 1: $b^2 - 4ac > 0$. Let k_1, k_2 be the distinct real roots. Then the general solution to the differential equation is $x(t) = c_1e^{k_1t} + c_2e^{k_2t}$.
   - Case 2: $b^2 - 4ac = 0$. Let k be the repeated root. Then the general solution to the differential equation is $x(t) = c_1e^{kt} + c_2te^{kt}$.
   - Case 3: $b^2 - 4ac < 0$. Let $k_1 = \alpha + i\beta, k_2 = \alpha - i\beta$ be the complex roots. Then the general solution to the differential equation is $x(t) = c_1e^{\alpha t}\cos(\beta t) + c_2e^{\alpha t}\sin(\beta t)$.

11.4.3 Nonhomogeneous Linear Equations and Undetermined Coefficients

This goal of this subsection is to discuss how to solve differential equations of the form $a_n x^{(n)} + a_{n-1}x^{(n-1)} + \cdots + a_1 x' + a_0 x = f(t)$ for some "nice" functions $f(t)$.

Any function x_p *that has no unknown parameters* and is a solution to $a_n x^{(n)} + a_{n-1}x^{(n-1)} + \cdots + a_1 x' + a_0 x = f(t)$ is called a **particular solution**. We should note that particular solutions are not unique.

Given a particular solution, x_p to $a_n x^{(n)} + a_{n-1}x^{(n-1)} + \cdots + a_1 x' + a_0 x = f(t)$ and the fundamental set of solutions, {x_1, x_2, \dots, x_n}, to the corresponding homogeneous differential equation $a_n x^{(n)} + a_{n-1}x^{(n-1)} + \cdots + a_1 x' + a_0 x = 0$, we can produce the general solution to $a_n x^{(n)} + a_{n-1}x^{(n-1)} + \cdots + a_1 x' + a_0 x = f(t)$. In particular, the general solution is $x(t) = c_1 x_1(t) + c_2 x_2(t) + \cdots + c_n x_n(t) + x_p(t)$ where $c_1, c_2, \dots, c_n$ are arbitrary constants. In other words, we simply need to add the solution to the associated homogeneous differential equation to a particular solution.

Motivating Example

Consider the second order linear differential equation $x'' + 2x' + x = e^{-5t}$. Find *one* solution.

First we will try to guess a particular solution. We know that $f(t) = e^{-5t}$ in this case. So we will guess a particular solution that looks similar, $x_p(t) = Ce^{-5t}$. Now we want to figure out what value of C (if any) makes $x_p(t)$ a solution

The method of undetermined coefficients is a technique used to solve certain types of non-homogeneous second-order differential equations. Specifically, this method is suitable for equations where the non-homogeneous term (the term not involving the function or its derivatives) is a simple function like a polynomial, exponential, sine, cosine, or a combination of these.

The overall process of the method of undetermined coefficients involves several steps:

1. Homogeneous Solution: First, solve the homogeneous version of the given differential equation (i.e., set the non-homogeneous term equal to zero). This gives the homogeneous solution, often denoted as x_h.
2. Particular Solution: Next, make an educated guess (using the table below) about the form of the particular solution x_p, which corresponds to the non-homogeneous term of the differential equation. The guessed form should match the non-homogeneous term but must include arbitrary constants (the "undetermined coefficients"). One important aspect to note is that if a term in the guessed x_p is already a solution to the corresponding homogeneous equation, this term must be multiplied by t to ensure linear independence. This is often referred to as the rule of "modification of trial functions".
3. Substitution: Substitute the guessed x_p and its derivatives into the original non-homogeneous differential equation. This will yield an equation in terms of the undetermined coefficients.
4. Determine Coefficients: Solve the resulting equation from the previous step to determine the values of the undetermined coefficients. **Note: The undetermine coefficients depend on the differential equation and not any initial conditions. This means that we can always solve for the undetermined coefficients.**
5. General Solution: The general solution to the original non-homogeneous differential equation is then the sum of the homogeneous solution x_h and the particular solution x_p.

While the method of undetermined coefficients is a powerful tool, it is not universal; its applicability is limited to differential equations with constant coefficients and fairly simple non-homogeneous terms. For more complex scenarios, other methods such as the method of variation of parameters may be needed.

$f(t)$	$x_p(t)$ (These are your guesses)
$a_n t^n + a_{n-1} t^{n-1} + \cdots + a_0$	$b_n t^n + b_{n-1} t^{n-1} + \cdots + b_0$
e^{at}	Ce^{at}
$\sin(bt)$	$c_1 \sin(bt) + c_2 \cos(bt)$
$\cos(bt)$	$c_1 \sin(bt) + c_2 \cos(bt)$
$e^t + t$	$Ce^t + (b_1 t + b_0)$
$t^2 \cos(t)$	$(b_2 x^2 + b_1 x + b_0)(c_1 \sin(bt) + c_2 \cos(bt))$

Example

Solve the initial value problem $x'' + 5x = -15$, $x(0) = 0$ and $x'(0) = 1$.

Step 1: Solve the homogeneous differential equation.

$$x'' + 5x = 0.$$

Thus, $k^2 + 5 = 0$ is the **characteristic equation** of this differential equation. Solving the characteristic equation gives $k = \pm\sqrt{5}i$. Thus, $x_h(t) = c_1 \cos(\sqrt{5}t) + c_2 \sin(\sqrt{5}t)$ is the general solution to the homogeneous differential equation.

Step 2: Guess x_p.

Note in this case $f(t) = -15$, which is a degree 1 polynomial. Using the table above our guess is $x_p(t) = b_0$. Notice that this guess for x_p is linearly independent from x_h so we do not need to adjust our guess.

Step 3: Solve the guess.

$$x_p(t) = b_0, \quad x_p'(t) = 0, \quad x_p''(t) = 0$$

Plugging this back into $x'' + 5x = -15$, we find that $0 + 5b_0 = -15$. Thus, $b_0 = -3$.

Step 4: General Solution

This means that $x(t) = x_h(t) + x_p(t) = c_1 \cos(\sqrt{5}t) + c_2 \sin(\sqrt{5}t) - 3$.

Step 5: Find c_1, c_2 using the initial conditions.

Since $x(0) = 0$, we have

$$0 = c_1 \cos(0) + c_2 \sin(0) - 3$$

$$0 = c_1 - 3$$

$$c_1 = 3.$$

Since $x'(t) = -\sqrt{5}c_1 \sin(\sqrt{5}t) + \sqrt{5}c_2 \cos(\sqrt{5}t)$ and $x'(0) = 1$, we have

$$1 = -\sqrt{5}c_1 \sin(0) + \sqrt{5}c_2 \cos(0)$$

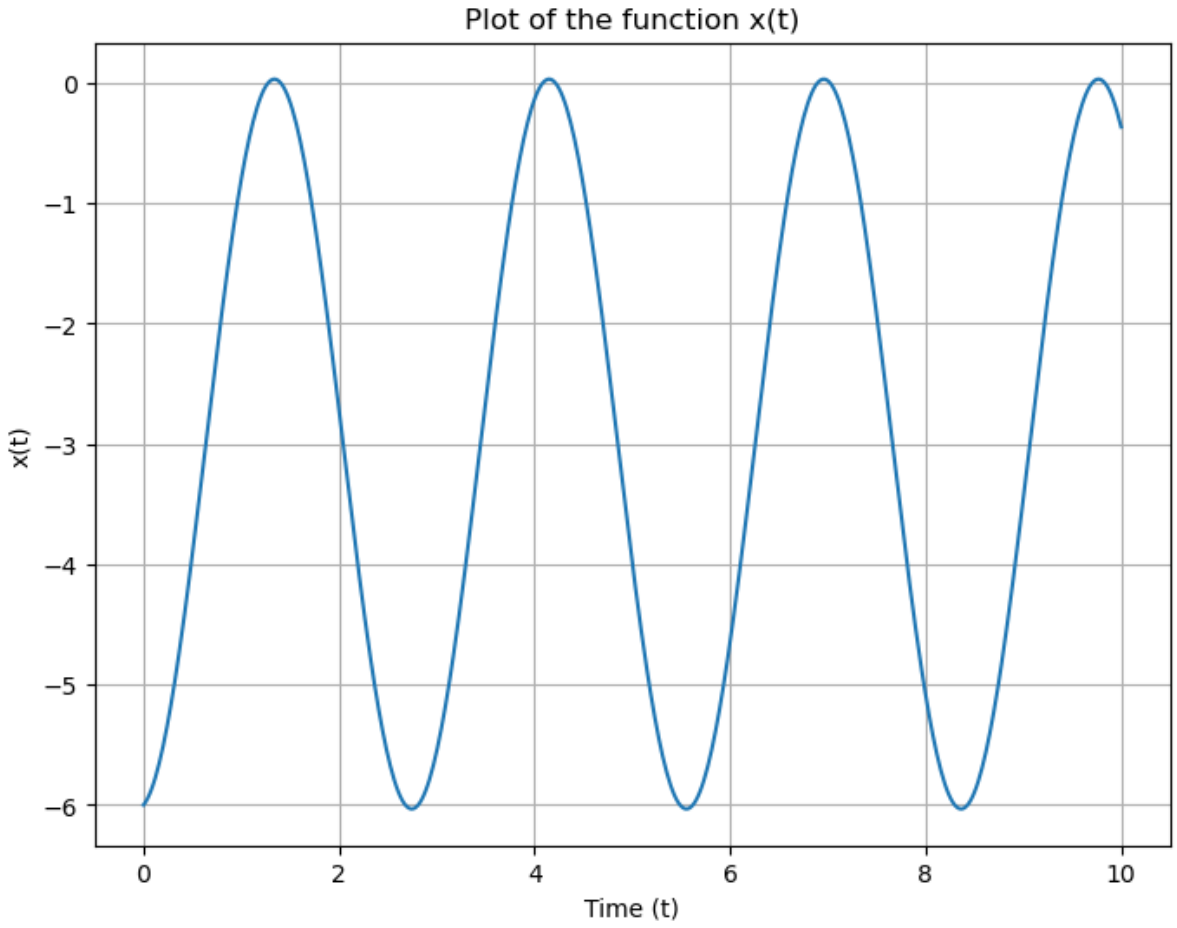
$$1 = \sqrt{5}c_2$$

$$c_2 = \frac{1}{\sqrt{5}}$$

Step 6: State the final solution.

This means the final solution to the initial value problem is $x(t) = 3\cos(\sqrt{5}t) + \frac{1}{\sqrt{5}} \sin(\sqrt{5}t) - 3$.

Now that we have the solution we can generate a plot of it to see how the solution behaves.



It turns out that the **Superposition Principle** holds for nonhomogeneous equations as well. In particular, suppose y_k is a particular solution to

$$a_n(x)y^{(n)}(x) + a_{n-1}y^{(n-1)}(x) + \cdots + a_1(x)y'(x) + a_0(x)y(x) = f_k(x).$$

Then the linear combination $y(x) = y_1(x) + y_2(x) + \cdots + y_n(x)$ is a particular solution to

$$a_n(x)y^{(n)}(x) + a_{n-1}y^{(n-1)}(x) + \cdots + a_1(x)y'(x) + a_0(x)y(x) = f_1(x) + f_2(x) + \cdots + f_n(x).$$

11.4.4 Exercises

Exercises

1. Find all solutions to $x'' + x' - 6x = 0$.
2. Use your answer to 1 to find the general solution to the differential equation $x'' + x' - 6x = \sin(3t)$.
3. Use your answer to 1 to find the general solution to the differential equation $x'' + x' - 6x = 5e^{4t}$.
4. Use your answer to 1 to find the general solution to the differential equation $x'' + x' - 6x = 5e^{2t}$. Hint: Think about why changing the exponent from $4t$ to $2t$ adds an extra step to the problem.
5. Use your answer to 2 and 3 to find the general solution of the differential equation $x'' + x' - 6x = 5e^{4t} + \sin(3t)$. Hint: What does the Superposition Principle tell us?

6. Find the form of the particular solution of the following differential equations. Note you do not need to solve for the undetermined coefficients.
   - $x'' - 6x' + 9x = 3t + 10\cos(t)$
   - $x'' - 9x = te^{3t}$
   - $2x'' + x' + 4x = 3t^2\sin(2t)$

11.5 Systems of First-Order Differential Equations

Note

This chapter offers an exploration into an essential component of the mathematical toolkit, used across numerous scientific and engineering disciplines. As we already know, differential equations are mathematical statements that describe the rates at which quantities change. However, reality often presents us with scenarios where multiple interrelated quantities are changing simultaneously. Enter the realm of systems of differential equations, a powerful extension of single differential equations, allowing us to model and understand a multitude of complex phenomena.

In this chapter, we specifically focus on systems of first-order differential equations. First-order means that only the first derivative of the function(s) appears in the system, but this doesn't imply simplicity. Even with this limitation, these systems can model a wide array of phenomena, from predator-prey relationships in ecology to interconnected circuits in electronics, oscillating chemical reactions, and the spread of diseases in populations, among many others.

11.5.1 Introduction to Systems of Linear Differential Equations

A **system of first-order differential equations** in n unknown functions $x_1(t), x_2(t), \ldots, x_n(t)$ is a collection of n differential equations with n dependent variables. That is,

$$\begin{aligned}
\frac{dx_1}{dt} &= f_1(t, x_1, x_2, \ldots, x_n) \\
\frac{dx_2}{dt} &= f_2(t, x_1, x_2, \ldots, x_n) \\
&\vdots \\
\frac{dx_n}{dt} &= f_n(t, x_1, x_2, \ldots, x_n).
\end{aligned}$$

In this case, a solution is a collection of n functions satisfying all of these equations. We should also note that every order n differential equation can be written as a system of n first-order differential equations.

Example

Write the differential equation $x''' + 4x'' + 2x' + x = t^2 + \cos(t)$ as a system of three first-order differential equations.

First we will define new variables. Let $x_1 = x, x_2 = x', x_3 = x''$. Then

$$x''' + 4x'' + 2x' + x = x_3' + 4x_3 + 2x_2 + x_1 = t^2 + \cos(t).$$

Rewriting this as a system yields

$$x_1' = x_2$$

$$x_2' = x_3$$

$$x_3' = -x_1 - 2x_2 - 4x_3 + t^2 + \cos(t)$$

Example

Find a system of first-order differential equations which are equivalent to the system

$$x'' = x + y\cos(t)$$

$$y'' = x\ln(t) + y'.$$

First we will define new variables. Let $x_1 = x, x_2 = x', y_1 = y, y_2 = y'$.

Rewriting this as a system yields

$$x_1' = x_2$$

$$x_2' = x_1 + y_1\cos(t)$$

$$y_1' = y_2$$

$$y_2' = x_1\ln(t) + y_2$$

A **system of linear first-order differential equations** (or simply **linear system** in n unknown functions $x_1(t), x_2(t), \ldots, x_n(t)$) is a system of the form

$$x_1' = a_{11}(t)x_1 + a_{12}(t)x_2 + \cdots + a_{1n}(t)x_n + b_1(t)$$

$$x_2' = a_{21}(t)x_1 + a_{22}(t)x_2 + \cdots + a_{2n}(t)x_n + b_2(t)$$

$$\vdots$$

$$x_n' = a_{n1}(t)x_1 + a_{n2}(t)x_2 + \cdots + a_{nn}(t)x_n + b_n(t)$$

where $a_{ij}(t)$ and $b_i(t)$ are arbitrary functions in the independent variable t. If $b_i(t) = 0$ for all $1 \leq i \leq n$, we say that the linear system is **homogeneous**. The number of dependent variables of the linear system is called the **dimension** of the system.

11.5.2 Solving Homogeneous Linear Systems with Constant Coefficients

The set $\{v_1(t), v_2(t), \ldots, v_n(t)\}$ where $v_i(t) = \begin{bmatrix} v_{i1}(t) \\ v_{i2}(t) \\ \vdots \\ v_{in}(t) \end{bmatrix}$ is called **linearly dependent** if there exist scalars $c_1, c_2, \ldots, c_n \in \mathbb{R}$, at least one of which is not zero, such that

$$c_1v_1(t) + c_2v_2(t) + \cdots + c_nv_n(t) = 0.$$

If no such scalars exist, the set of vector functions is **linearly independent**.

Note: We saw the same notion for functions and vectors when looking at solving linear (usually second order) homogeneous differential equations with constant coefficients.

Example

- $\left\{\begin{bmatrix} e^{2t} \\ e^{3t} \end{bmatrix}, \begin{bmatrix} e^{3t} \\ e^{2t} \end{bmatrix}\right\}$ are linearly independent.
- $\left\{\begin{bmatrix} e^{2t} \\ e^{3t} \end{bmatrix}, \begin{bmatrix} 4e^{2t} \\ 4e^{3t} \end{bmatrix}\right\}$ are linearly dependent as $4\begin{bmatrix} e^{2t} \\ e^{3t} \end{bmatrix} - \begin{bmatrix} 4e^{2t} \\ 4e^{3t} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$
- $\left\{v_1(t) = \begin{bmatrix} \sin(t) \\ t \end{bmatrix}, v_2(t) = \begin{bmatrix} \sin(t) \\ 3t \end{bmatrix}, v_3(t) = \begin{bmatrix} 0 \\ -2t \end{bmatrix}\right\}$ are linearly dependent as $v_1(t) - v_2(t) - v_3(t) = \vec{0}$.

If $x' = Ax$ is a homogeneous linear system of first-order differential equations with $n \times n$ coefficient matrix A, any set of n linearly independent vector solutions to the system on a given interval I is called a **fundamental set of solutions** to the system. If $\{v_1(t), v_2(t), \ldots, v_n(t)\}$ is a fundamental set of solutions, then

$$x(t) = c_1 v_1(t) + c_2 v_2(t) + \cdots + c_n v_n(t)$$

is called the **general solution**.

Example

Consider the system

$$x' = \begin{bmatrix} 5 & -4 \\ 2 & -1 \end{bmatrix} x.$$

Verify that

$$\left\{\begin{bmatrix} 2e^{3t} \\ e^{3t} \end{bmatrix}, \begin{bmatrix} e^{t} \\ e^{t} \end{bmatrix}\right\}$$

forms a fundamental solution set.

Step 1: Substituting in both vectors from the fundamental solution set to the left hand side yields

$$\frac{d}{dt}\left(\begin{bmatrix} 2e^{3t} \\ e^{3t} \end{bmatrix}\right) = \begin{bmatrix} 6e^{3t} \\ 3^{3t} \end{bmatrix}$$

and

$$\frac{d}{dt}\left(\begin{bmatrix} e^{t} \\ e^{t} \end{bmatrix}\right) = \begin{bmatrix} e^{t} \\ e^{t} \end{bmatrix}.$$

Step 2: Substituting in both vectors from the fundamental solution set to the right hand side yields

$$\begin{bmatrix} 5 & -4 \\ 2 & -1 \end{bmatrix}\begin{bmatrix} 2e^{3t} \\ e^{3t} \end{bmatrix} = \begin{bmatrix} 10e^{3t} - 4e^{3t} \\ 4e^{3t} - e^{3t} \end{bmatrix} = \begin{bmatrix} 6e^{3t} \\ 3e^{3t} \end{bmatrix}$$

and

$$\begin{bmatrix} 5 & -4 \\ 2 & -1 \end{bmatrix}\begin{bmatrix} e^{t} \\ e^{3t} \end{bmatrix} = \begin{bmatrix} 5e^{t} - 4e^{t} \\ 2e^{t} - e^{t} \end{bmatrix} = \begin{bmatrix} e^{t} \\ e^{t} \end{bmatrix},$$

Step 3: Notice that both vectors are solutions to the system. Now to verify that they form a fundamental set of solutions, we simply need to check that the two vectors are linearly independent. Solving the system

$$2c_1 e^{3t} + c_2 e^{t} = 0$$

$$c_1 e^{3t} + c_2 e^t = 0$$

gives that $c_1 = c_2 = 0$. Thus the vectors are linearly independent.

This means the general solution to this system is

$$x(t) = c_1 \begin{bmatrix} 2 \\ 1 \end{bmatrix} e^{3t} + c_2 \begin{bmatrix} 1 \\ 1 \end{bmatrix} e^t.$$

Natural Question

Where do the functions e^t and e^{3t} and the vectors $\begin{bmatrix} 2 \\ 1 \end{bmatrix}$ and $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ come from?

It turns out that these values come from the eigenvalues and eigenvectors of $\begin{bmatrix} 5 & -4 \\ 2 & -1 \end{bmatrix}$. So let's compute those values!

$$\det(A_\lambda I) = 0$$

$$\det\left(\begin{bmatrix} 5-\lambda & -4 \\ 2 & -1-\lambda \end{bmatrix}\right) = 0$$

$$(5-\lambda)(-1-\lambda) - (-8) = 0$$

$$\lambda^2 - 4\lambda + 3 = 0$$

$$(\lambda - 3)(\lambda - 1) = 0$$

Thus the eigenvalues are $\lambda = 1$ and $\lambda = 3$.

Now we want to find the eigenvectors associated to each eigenvalue. Let's start with $\lambda = 1$.

Setting up the augmented matrix yields

$$\left[\begin{array}{cc|c} 5-1 & -4 & 0 \\ 2 & -1-1 & 0 \end{array}\right] = \left[\begin{array}{cc|c} 4 & -4 & 0 \\ 2 & -2 & 0 \end{array}\right] \approx \left[\begin{array}{cc|c} 1 & -1 & 0 \\ 0 & 0 & 0 \end{array}\right]$$

Thus an eigenvector associated to $\lambda = 1$ is $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

Let's now consider $\lambda = 3$.

Setting up the augmented matrix yields

$$\left[\begin{array}{cc|c} 2 & -4 & 0 \\ 2 & -4 & 0 \end{array}\right] \approx \left[\begin{array}{cc|c} 1 & -2 & 0 \\ 0 & 0 & 0 \end{array}\right]$$

Thus an eigenvector associated to $\lambda = 3$ is $\begin{bmatrix} 2 \\ 1 \end{bmatrix}$.

Notice that these values recover the information from the fundamental solution set in the previous example.

Theorem

Let A be an $n \times n$ matrix of real numbers with n distinct real eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ with respective eigenvectors $v_1, v_2, \ldots, v_n$. Then

$$\{v_1 e^{\lambda_1 t}, v_2 e^{\lambda_2 t}, \ldots, v_n e^{\lambda_n t}\}$$

is a fundamental set of solutions to the homogeneous linear system $x' = Ax$.

Example

Solve the initial-value problem

$$x' = \begin{bmatrix} 5 & 3 \\ 6 & 2 \end{bmatrix} x, \quad x(0) = \begin{bmatrix} 0 \\ 3 \end{bmatrix}.$$

$$\det(A - \lambda I) = 0$$

$$\det\left(\begin{bmatrix} 5-\lambda & 3 \\ 6 & 2-\lambda \end{bmatrix}\right) = 0$$

$$(5-\lambda)(2-\lambda) - (18) = 0$$

$$\lambda^2 - 7\lambda - 8 = 0$$

$$(\lambda + 1)(\lambda - 8) = 0$$

Thus the eigenvalues are $\lambda = -1$ and $\lambda = 8$.

Now we want to find the eigenvectors associated to each eigenvalue. Let's start with $\lambda = -1$.

Setting up the augmented matrix yields

$$\begin{bmatrix} 5-(-1) & 3 \\ 6 & 2-(-1) \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \left[\begin{array}{cc|c} 6 & 3 & 0 \\ 6 & 3 & 0 \end{array}\right] \approx \left[\begin{array}{cc|c} 1 & \frac{1}{2} & 0 \\ 0 & 0 & 0 \end{array}\right]$$

Thus an eigenvector associated to $\lambda = -1$ is $\begin{bmatrix} \frac{-1}{2} \\ 1 \end{bmatrix}$.

Let's now consider $\lambda = 8$.

Setting up the augmented matrix yields

$$\left[\begin{array}{cc|c} -3 & 3 & 0 \\ 6 & -6 & 0 \end{array}\right] \approx \left[\begin{array}{cc|c} 1 & -1 & 0 \\ 0 & 0 & 0 \end{array}\right]$$

Thus an eigenvector associated to $\lambda = 8$ is $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

This means the general solution to this system is $x(t) = c_1 \begin{bmatrix} \frac{-1}{2} \\ 1 \end{bmatrix} e^{-t} + c_2 \begin{bmatrix} 1 \\ 1 \end{bmatrix} e^{8t}$.

Using the initial condition, we find that

$$\begin{bmatrix} 0 \\ 3 \end{bmatrix} = c_1 \begin{bmatrix} \frac{-1}{2} \\ 1 \end{bmatrix} e^{-(0)} + c_2 \begin{bmatrix} 1 \\ 1 \end{bmatrix} e^{8(0)}$$

$$\begin{bmatrix} 0 \\ 3 \end{bmatrix} = c_1 \begin{bmatrix} \frac{-1}{2} \\ 1 \end{bmatrix} + c_2 \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

This is equivalent to solving

$$0 = \frac{-1}{2} c_1 + c_2$$

$$3 = c_1 + c_2.$$

Solving this system for c_1 and c_2 gives $c_1 = 2$ and $c_2 = 1$.

Thus the solution to the initial value problem is

$$x(t) = 2\begin{bmatrix} \frac{-1}{2} \\ 1 \end{bmatrix} e^{-t} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} e^{8t}.$$

Now let's consider what happens when we do not have n distinct eigenvalues. In other words, what happens when an eigenvalue occurs with some multiplicity.

Motivating Example

Consider the following matrices

$$B = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \qquad C = \begin{bmatrix} 3 & 1 \\ -1 & 1 \end{bmatrix}.$$

Both of these matrices have $\lambda = 2$ as an eigenvalue with multiplicity 2.

In the case of matrix B, $\lambda = 2$ has two linearly independent eigenvectors $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$. In this case, the procedure for finding the general solution to the equation $x' = Bx$ has not changed from the previous examples.

However, in the case of matrix C, $\lambda = 2$ has only one eigenvector $\begin{bmatrix} -1 \\ 1 \end{bmatrix}$ despite having multiplicity 2. In this case, the procedure for finding the general solution to the equation $x' = Cx$ will need to be changed as we do not have enough linearly independent pieces for our fundamental solution set.

Let A be an $n \times n$ matrix with eigenvalue λ appearing with multiplicity k in the characteristic polynomial. The eigenvalue λ has at least 1 and at most k linearly independent eigenvectors. The eigenvalue λ is called **defective** if there are fewer than k linearly independent eigenvectors.

There are a few significant observations we should make about this example and subsequent definition. First, if a matrix has n distinct eigenvalues, none of the eigenvalues will be defective. Second, if λ is not defective, then the initial procedure we establish for distinct real eigenvalues will still work. Finally, if λ is defective, we will need to produce at least one (if not more) additional solution to create the full fundamental solution set.

Example

Solve the linear system of differential equations

$$x' = \begin{bmatrix} 3 & 1 \\ -1 & 1 \end{bmatrix} x.$$

$$\det(C - \lambda I) = 0$$

$$\det\left(\begin{bmatrix} 3-\lambda & 1 \\ -1 & 1-\lambda \end{bmatrix}\right) = 0$$

$$(3-\lambda)(1-\lambda) - (-1) = 0$$

$$\lambda^2 - 4\lambda + 4 = 0$$

$$(\lambda - 2)^2 = 0$$

Thus the eigenvalue is $\lambda = 2$ with multiplicity 2.

Now we want to find the eigenvector(s) associated with $\lambda = 2$.

Setting up the augmented matrix yields

$$\left[\begin{array}{cc|c} 3-(2) & 1 & 0 \\ -1 & 1-(2) & 0 \end{array}\right] = \left[\begin{array}{cc|c} 1 & 1 & 0 \\ -1 & -1 & 0 \end{array}\right] \approx \left[\begin{array}{cc|c} 1 & 1 & 0 \\ 0 & 0 & 0 \end{array}\right]$$

Thus the eigenvector associated to $\lambda = 2$ is $\begin{bmatrix} -1 \\ 1 \end{bmatrix}$.

Let's now guess another solution. The guess is $\begin{bmatrix} -1 \\ 1 \end{bmatrix} te^{2t}$.

Notice

$$C \begin{bmatrix} -1 \\ 1 \end{bmatrix} te^{2t} = \begin{bmatrix} 3 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} te^{2t} = \begin{bmatrix} -2 \\ 2 \end{bmatrix} te^{2t}$$

and

$$\frac{d}{dt}\left(\begin{bmatrix} -1 \\ 1 \end{bmatrix} te^{2t}\right) = \begin{bmatrix} -1 \\ 1 \end{bmatrix} e^{2t} + \begin{bmatrix} -2 \\ 2 \end{bmatrix} te^{2t}.$$

Notice these are not quite equal, but they are close!

Let's make a new guess. The guess is $\begin{bmatrix} -1 \\ 1 \end{bmatrix} te^{2t} + \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} e^{2t}$ where $\begin{bmatrix} 3-2 & 1 \\ -1 & 1-2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$. We can verify that this new guess is in fact a solution!

This means the general solution to this system is $x(t) = c_1 \begin{bmatrix} -1 \\ 1 \end{bmatrix} e^{2t} + c_2 \left(\begin{bmatrix} -1 \\ 1 \end{bmatrix} te^{2t} + \begin{bmatrix} -2 \\ 1 \end{bmatrix} e^{2t} \right)$.

Theorem

If $x' = Ax$ is a homogeneous linear system and λ is a defective eigenvalue of multiplicity 2 with corresponding eigenvalue v, then

$$vte^{\lambda t} + we^{\lambda t}$$

where w is any solution to

$$(A - \lambda I)w = v$$

is another solution to $x' = Ax$. Moreover, this solution is linearly independent to the other solution associated with λ by construction.

Now let's consider what happens when we have complex eigenvalues. It turns out that in this case, we have a nice procedure we can follow. Suppose A has eigenvalues $\lambda = a \pm bi$. Then we can produce real valued solutions to $x' = Ax$ associated to these eigenvalues as follows:

1. Choose **one** of the eigenvalues $\lambda = a \pm bi$.
2. Find an eigenvector v associated to λ.
3. Rewrite the solution $x(t) = ve^{\lambda t} = \text{Re}(x(t)) + i\text{Im}(x(t))$ using Euler's formula and algebra.

4. $\text{Re}(x(t))$ and $\text{Im}(x(t))$ are real-valued solutions to the linear system of differential equations so we can add them to the fundamental solution set.

Now let's see this procedure applied to an example.

Example

Solve the linear system of differential equations

$$x' = \begin{bmatrix} 4 & -10 \\ 2 & -4 \end{bmatrix} x.$$

$$\det(A - \lambda I) = 0$$

$$\det\left(\begin{bmatrix} 4-\lambda & -10 \\ 2 & -4-\lambda \end{bmatrix}\right) = 0$$

$$(4-\lambda)(-4-\lambda) + (20) = 0$$

$$\lambda^2 + 4 = 0$$

Thus the eigenvalues are $\lambda = \pm 2i$.

Now we want to find the eigenvector associated with $\lambda = -2i$.

Setting up the augmented matrix yields

$$\begin{bmatrix} 4-(-2i) & -10 \\ 2 & -4-(-2i) \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} \approx \begin{bmatrix} 1 & -2+i \\ 2 & -4+2i \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} \approx \begin{bmatrix} 1 & -2+i \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Thus the eigenvector associated to $\lambda = -2i$ is $\begin{bmatrix} 2-i \\ 1 \end{bmatrix}$.

We also know that the eigenvector associated to $\lambda = 2i$ is the complex conjugate of $\begin{bmatrix} 2-i \\ 1 \end{bmatrix}$. Thus the eigenvector associated to $\lambda = 2i$ is $\begin{bmatrix} 2+i \\ 1 \end{bmatrix}$.

This means that a fundamental solution set is $\left\{ \begin{bmatrix} 2-i \\ 1 \end{bmatrix} e^{-2it}, \begin{bmatrix} 2+i \\ 1 \end{bmatrix} e^{2it} \right\}$. But if we want a real-valued solution (which is better for interpreting physical situations), we need to find a different fundamental solution set using the process outlined above.

Step 1 & Step 2: Let $x_1(t) = \begin{bmatrix} 2+i \\ 1 \end{bmatrix} e^{2it}$.

Step 3: Using Euler's formula and algebra $x_1(t) = \begin{bmatrix} 2+i \\ 1 \end{bmatrix} (\cos(2t) + i\sin(2t)) = \begin{bmatrix} 2\cos(2t) - \sin(2t) \\ \cos(2t) \end{bmatrix} + i \begin{bmatrix} \cos(2t) + 2\sin(2t) \\ \sin(2t) \end{bmatrix}$.

Step 4: Create the new real-valued fundamental solution set from Step 3. The new fundamental solution set is $\{\text{Re}(x(t)), \text{Im}(x(t))\} = \left\{ \begin{bmatrix} 2\cos(2t) - \sin(2t) \\ \cos(2t) \end{bmatrix}, \begin{bmatrix} \cos(2t) + 2\sin(2t) \\ \sin(2t) \end{bmatrix} \right\}$.

This means that the general solution to this system of differential equations is

$$x(t) = c_1 \begin{bmatrix} 2\cos(2t) - \sin(2t) \\ \cos(2t) \end{bmatrix} + c_2 \begin{bmatrix} \cos(2t) + 2\sin(2t) \\ \sin(2t) \end{bmatrix}.$$

11.5.3 Solving Non-Homogeneous Linear Systems with Constant Coefficients

In Section 5.2, we discussed how to solve homogeneous linear systems with constant coefficients. What if we now wanted to solve non-homogeneous linear systems of differential equations? It turns out that there is a nice procedure for solving these. Let $x' = Ax + b$ be a non-homogeneous linear system of differential equations. Then

1. Find the **complementary function** x_c, which is the general solution of the associated linear system $x' = Ax$.
2. Use the **method of undetermined coefficients** (i.e. guess a solution and solve for the coefficients) to produce a particular solution x_p of the original, non-homogeneous system.
3. Then the general solution $x = x_c + x_p$.
4. If initial conditions are given, use the general solution to and initial conditions to solve the initial value problem. *Note: This step must be done last.*

Prior to looking at an example, let's see how we make the guess in Step 2. The goal is to "match" the form of the non-homogeneous part with our guess.

$\vec{f}(t)$	$\vec{x}_p(t)$ (These are your guesses)
$\vec{a}_n t^n + \vec{a}_{n-1} t^{n-1} + \cdots + \vec{a}_0$	$\vec{b}_n t^n + \vec{b}_{n-1} t^{n-1} + \cdots + \vec{b}_0$
$\vec{b} e^{kt}$	$\vec{c} e^{kt}$
$\vec{b} \sin(kt)$	$\vec{c} \sin(kt) + \vec{d} \cos(kt)$
$\vec{b} \cos(kt)$	$\vec{c} \sin(kt) + \vec{d} \cos(kt)$

Example

Solve the non-homogeneous linear system of differential equations

$$x' = \begin{bmatrix} 3 & 1 \\ -1 & 1 \end{bmatrix} x + \begin{bmatrix} 4 \\ 4t \end{bmatrix}.$$

Step 1: Solve the homogeneous linear system $x' = \begin{bmatrix} 3 & 1 \\ -1 & 1 \end{bmatrix} x$.

Notice we already solved this in Section 5.2. The general solution to the associated homogeneous system is

$$x_c(t) = c_1 \begin{bmatrix} -1 \\ 1 \end{bmatrix} e^{2t} + c_2 \left(\begin{bmatrix} -1 \\ 1 \end{bmatrix} te^{2t} + \begin{bmatrix} -2 \\ 1 \end{bmatrix} e^{2t} \right).$$

Step 2: Guess a particular solution $x_p(t) = \begin{bmatrix} a + bt \\ c + dt \end{bmatrix}$ and solve for the values of a, b, c, d. Notice that 4 is a degree 0 polynomial and $4t$ is a degree 1 polynomial so for our guess, we guess an arbitrary degree 1 polynomial for each coordinate. Notice

$$\frac{d}{dt} \begin{bmatrix} a + bt \\ c + dt \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix}$$

and

$$\begin{bmatrix} 3 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} a + bt \\ c + dt \end{bmatrix} + \begin{bmatrix} 4 \\ 4t \end{bmatrix} = \begin{bmatrix} 3a + 3bt + c + dt + 4 \\ -a - bt + c + dt + 4t \end{bmatrix}.$$

Setting the left hand side and right hand side equal to each other yields

$$3a + 3bt + c + dt + 4 = b$$

and

$$-a - bt + c + dt + 4t = d.$$

This means that

$$3a - b + c = -4$$

$$3b + d = 0$$

$$-a + c - d = 0$$

$$-b + d = 4.$$

When we solve for a, b, c, d (using techniques found in the Linear Algebra chapter), we find that $a = 0$, $b = 1$, $c = -3$, and $d = -3$. This mean that the general solution to this non-homogeneous linear system is

$$x(t) = c_1 \begin{bmatrix} -1 \\ 1 \end{bmatrix} e^{2t} + c_2 \left(\begin{bmatrix} -1 \\ 1 \end{bmatrix} te^{2t} + \begin{bmatrix} -2 \\ 1 \end{bmatrix} e^{2t} \right) + \begin{bmatrix} t \\ -3 - 3t \end{bmatrix}.$$

11.5.4 The Phase Plane for Autonomous Systems

A system of two first-order differential equations is called **autonomous** if it is of the form $\frac{dx}{dt} = f(x, y)$ $\frac{dy}{dt} = g(x, y)$. In other words, it is called autonomous if only the dependent variables of the system appear in either equation.

Note: This definition can be extended to systems of dimension > 2 in the natural way. However, we can understand the geometry of autonomous systems of dimension 2 very well so that is where the focus will be.

Example

- $\begin{cases} x' = x^2y + y - x \\ y' = x + y - xy \end{cases}$ is an autonomous system.
- $\begin{cases} x' = x^2y + ty - x \\ y' = x + ty - t^2 \end{cases}$ is not an autonomous system because the independent variable t appears.

A **phase plane** for an autonomous system

$$\frac{dx}{dt} = f(x, y)$$

$$\frac{dy}{dt} = g(x, y)$$

is the grid $\mathbb{R}^2$ in which vectors of the form

$$\vec{v} = x'(t)\vec{i} + y'(t)\vec{j} = \langle f(x, y), g(x, y) \rangle$$

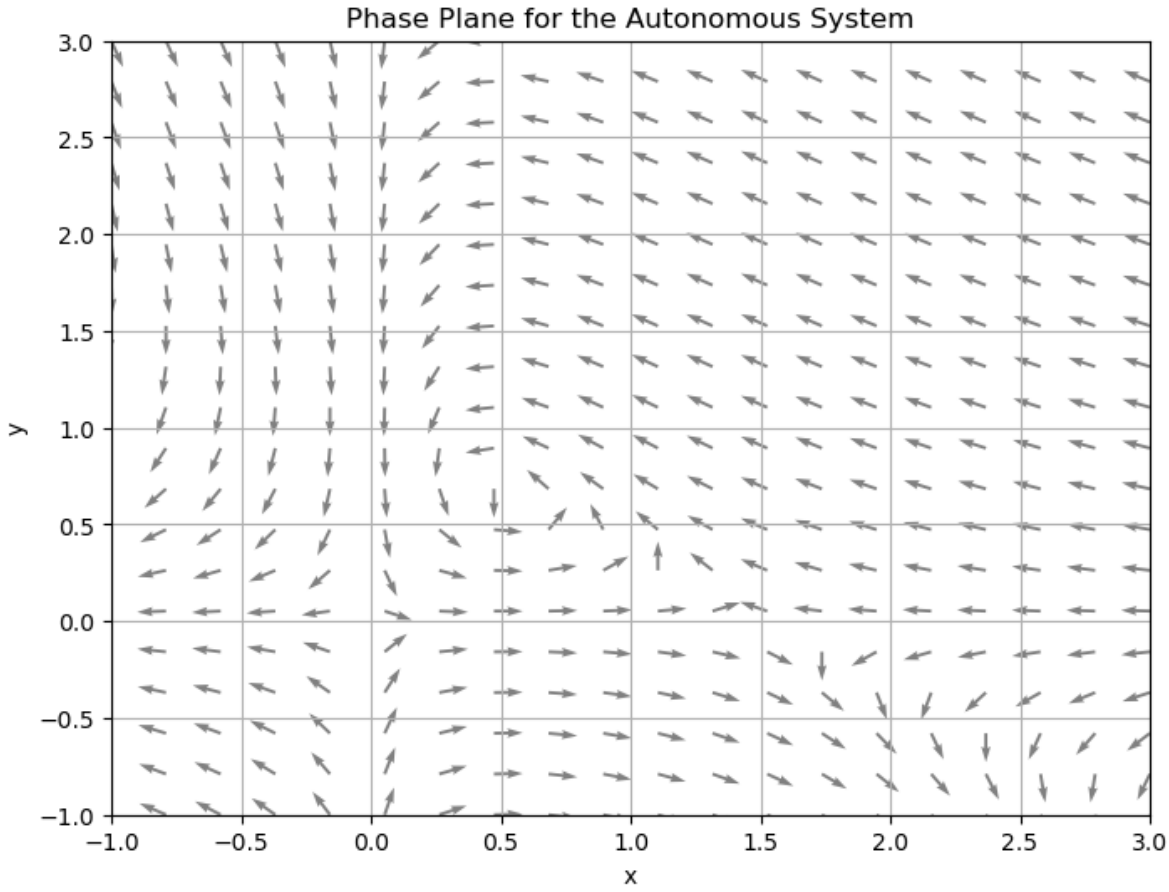
have been drawn at a set of grid points (x_i, y_i). A **phase portrait** for the autonomous system is a phase plane with enough solution curves drawn in to show how solutions behave in every part of the plane.

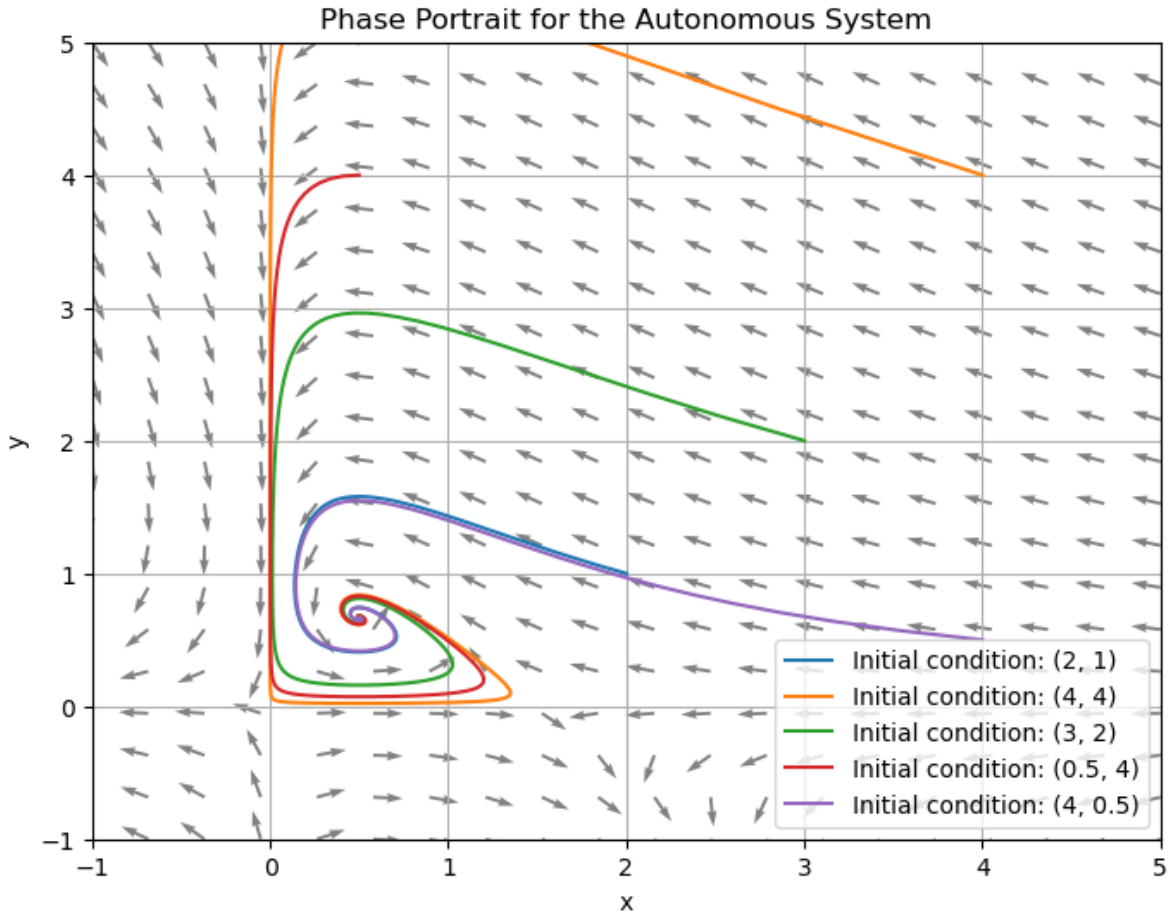
Example

Draw a phase plane and then a phase portrait for the system

$$x' = x(3 - 2x) - 3xy$$

$$y' = -y + 2xy.$$





An **equilibrium solution** of an autonomous system

$$\frac{dx}{dt} = f(x, y)$$

$$\frac{dy}{dt} = g(x, y)$$

is a constant solution $x(t) = \bar{x}$, $y(t) = \bar{y}$ for which $f(\bar{x}, \bar{y})$ and $g(\bar{x}, \bar{y})$ are simultaneously 0. The point in the phase plane corresponding to an equilibrium solution is called an equilibrium point.

Example

Find the equilibrium solutions of the system and plot those points in the phase plane

$$x' = x(3 - 2x) - 3xy$$

$$y' = -y + 2xy.$$

To find the equilibrium solutions, we need to solve the system

$$3x - 2x^2 - 3xy = 0 \rightarrow x(3 - 2x - 3y) = 0$$

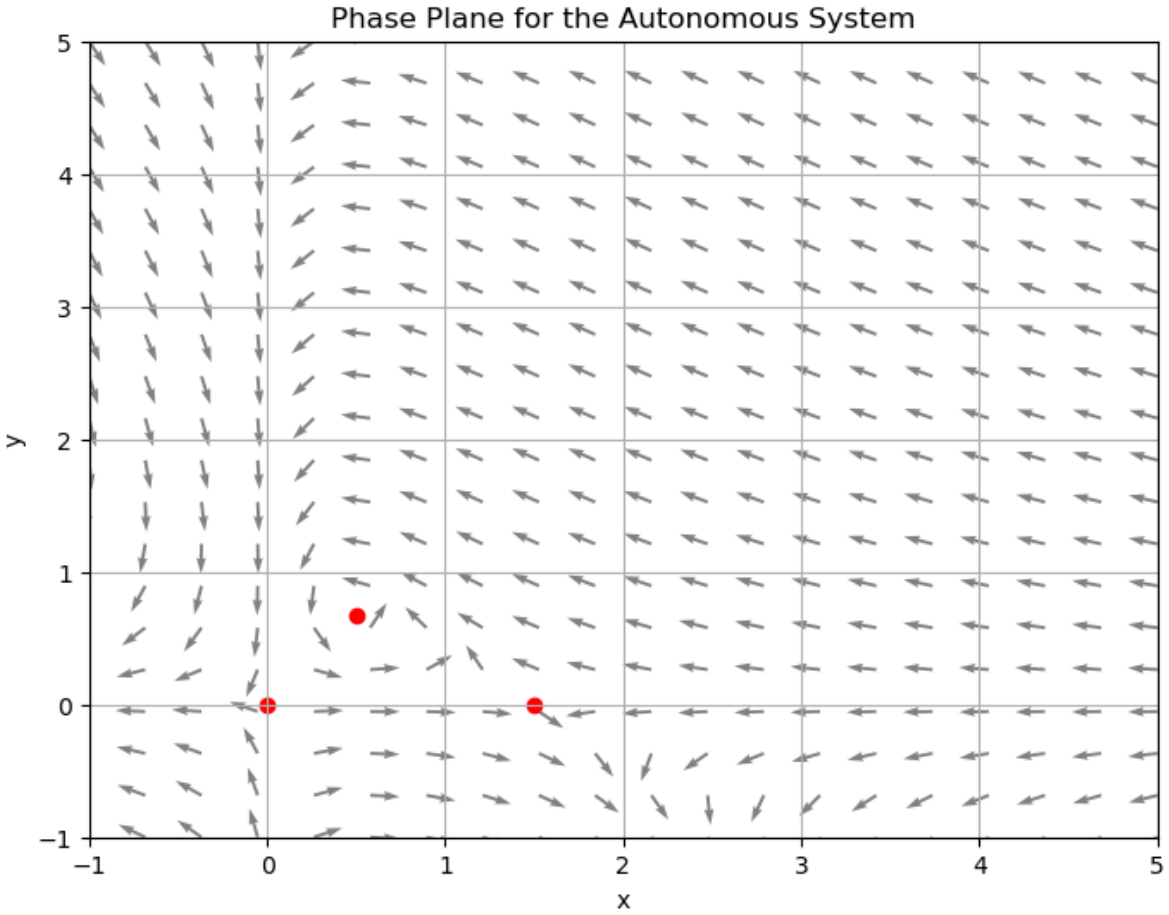
$$-y + 2xy = 0 \rightarrow y(-1 + 2x) = 0$$

This tells us that either $x = 0$ OR $3 - 2x - 3y = 0$ or $y = 0$ OR $-1 + 2x = 0$.

If $x = 0$, then $y = 0$. So $(0, 0)$ is an equilibrium solution.

If $y = 0$ and $x \neq 0$, then $3 - 2x - 3(0) = 0 \rightarrow x = \frac{3}{2}$. So $\left(\frac{3}{2}, 0\right)$ is an equilibrium solution.

If $-1 + 2x = 0 \rightarrow x = \frac{1}{2}$, then $3 - 2\left(\frac{1}{2}\right) - 3y = 0 \rightarrow y = \frac{2}{3}$. So $\left(\frac{1}{2}, \frac{2}{3}\right)$ is the third and final equilibrium solution.



11.5.5 Geometric Behavior of Homogeneous Linear Systems

Let A be a 2×2 matrix where A has two distinct real eigenvalues and consider the system $x' = Ax$:

- $x' = Ax$ has a unique solution for any initial condition $(x_0, y_0) \in \mathbb{R}^2$
- If $\det(A) \neq 0$, then $(0, 0)$ is the only equilibrium solution of $x' = Ax$
- The lines spanned by the eigenvectors form straight line solution curves
- For large values of t, solutions will tend towards the straight line solution for the largest eigenvalue (sometimes called the **dominant eigenvalue**)

Subsequently, to describe the geometric behavior of the solutions, it is enough to draw the eigenvalue solutions, and one sample solution curve in each of the four regions.

Example

Consider the two-dimensional homogeneous linear system

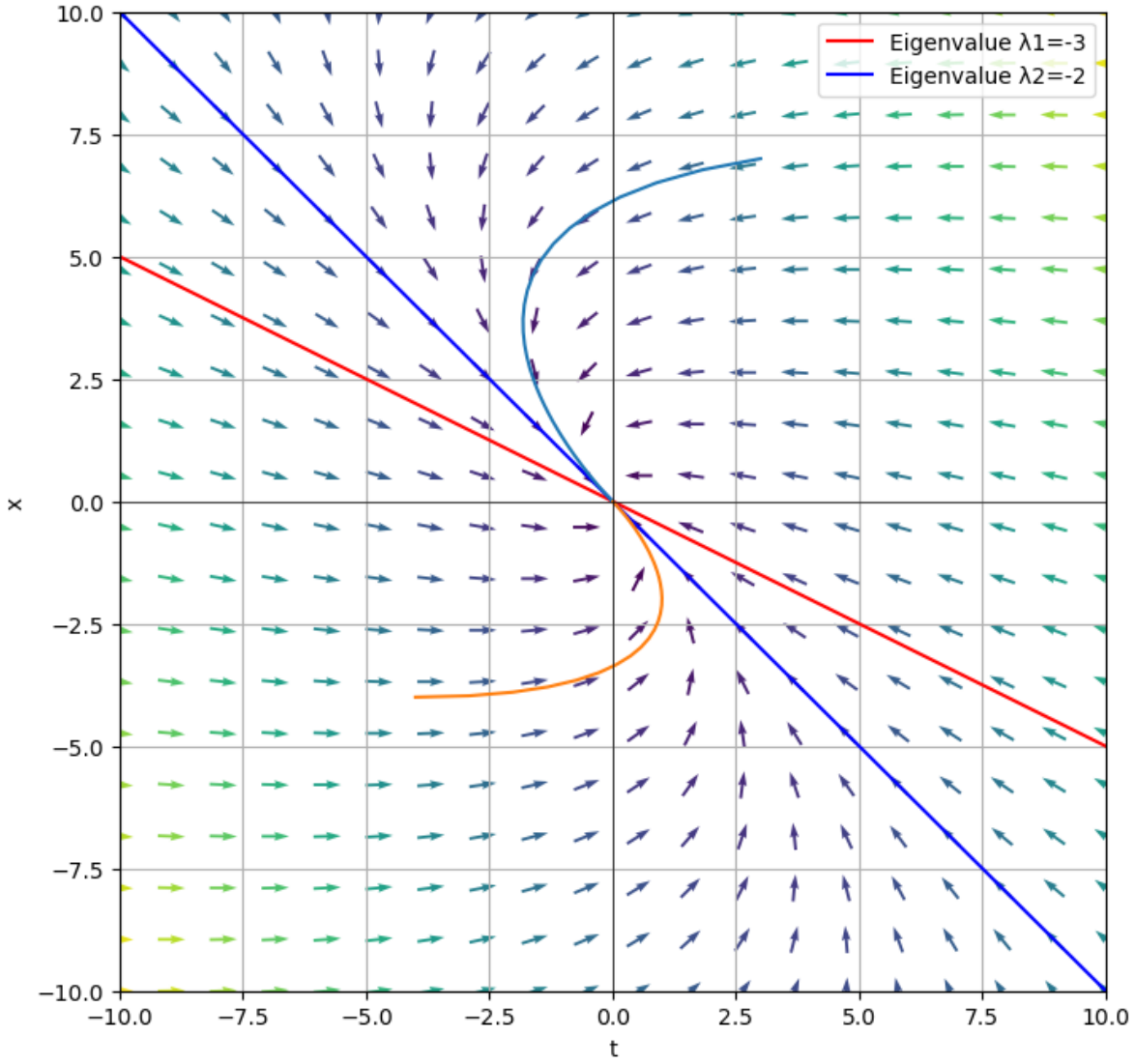
$$x' = \begin{bmatrix} -4 & -2 \\ 1 & -1 \end{bmatrix} x$$

which has eigenvalues $\lambda_1 = -3$ and $\lambda_2 = -2$ with respective eigenvectors

$$v_1 = \begin{bmatrix} -2 \\ 1 \end{bmatrix}$$

and

$$v_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$



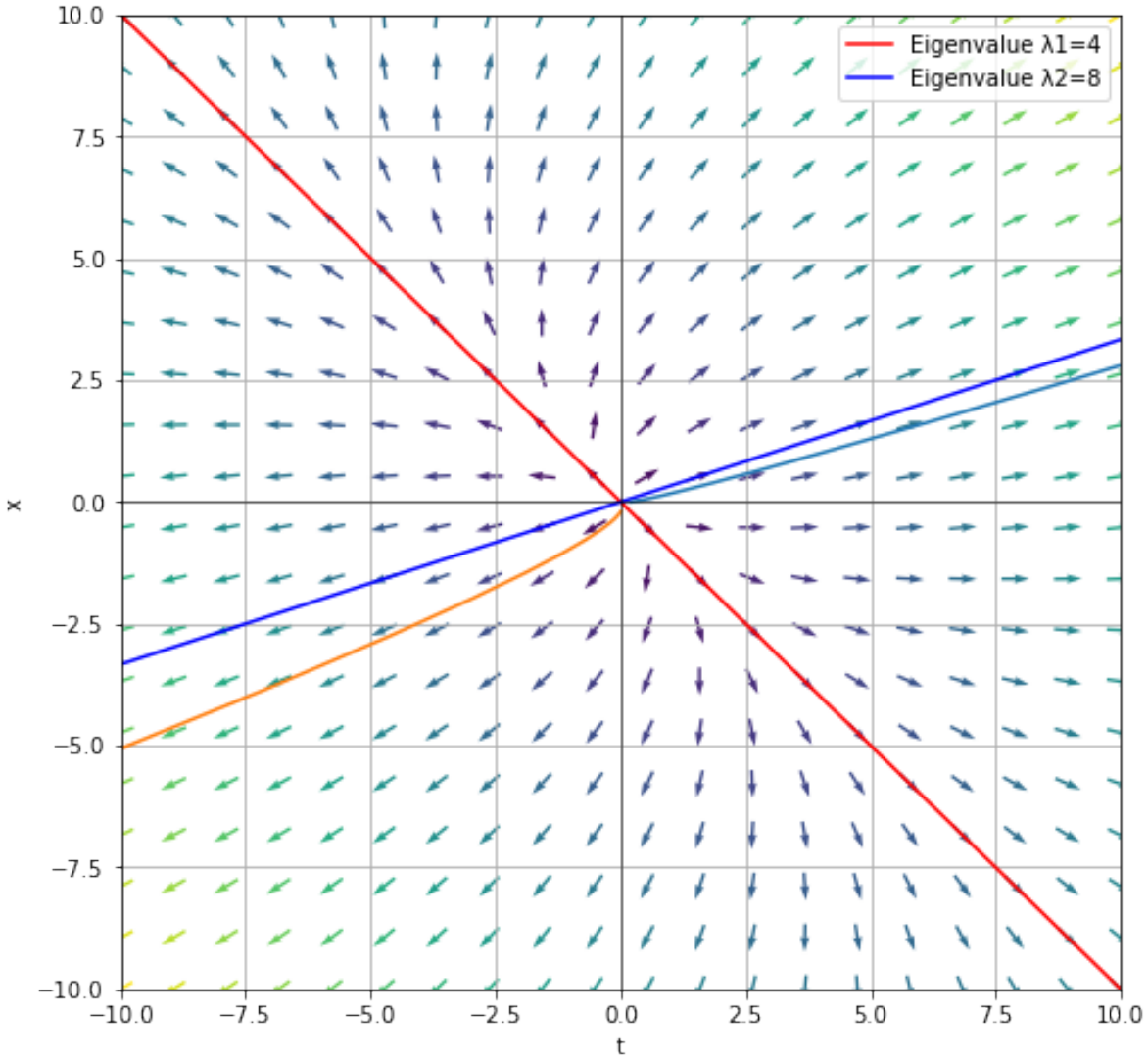
Notice that we can see from the phase plane that $(0, 0)$ is a stable equilibrium. Also notice that as $t \to \infty$, the solutions approach the stable equilibrium.

Example

Consider the two-dimensional homogeneous linear system

$$x' = \begin{bmatrix} 7 & 3 \\ 1 & 5 \end{bmatrix} x$$

which has eigenvalues $\lambda_1 = 4$ and $\lambda_2 = 8$ with respective eigenvectors $v_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ and $v_2 = \begin{bmatrix} 1 \\ \frac{1}{3} \end{bmatrix}$.



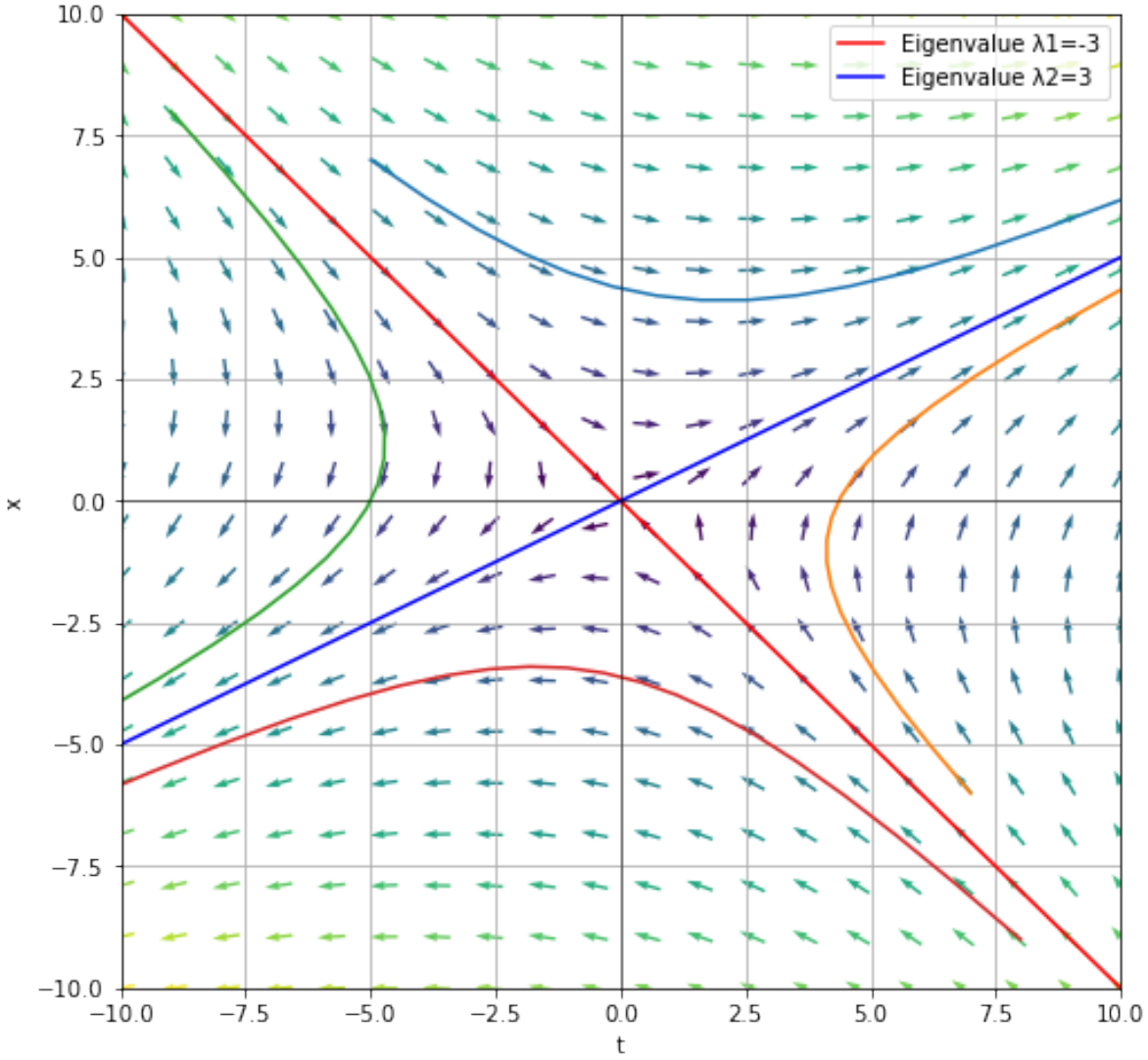
Notice that we can see from the phase plane that $(0, 0)$ is an unstable equilibrium. Also notice that as $t \to \infty$, the solutions approach $\begin{bmatrix} 1 \\ \frac{1}{3} \end{bmatrix} e^{8t}$.

Example

Consider the two-dimensional homogeneous linear system

$$x' = \begin{bmatrix} 1 & 4 \\ 2 & -1 \end{bmatrix} x$$

which has eigenvalues $\lambda_1 = -3$ and $\lambda_2 = 3$ with respective eigenvectors $v_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ and $v_2 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$.



Notice that we can see from the phase plane that $(0, 0)$ is a semi-stable equilibrium. In particular, if our initial condition started on $\begin{bmatrix} -1 \\ 1 \end{bmatrix} e^{-3t}$, then it will tend towards zero. But if the initial condition does not lie on that line, then the solutions tends towards $\begin{bmatrix} 2 \\ 1 \end{bmatrix} e^{3t}$.

Let A be a 2×2 matrix where A has two complex eigenvalues, $\lambda = a \pm bi$, and consider the system $x' = Ax$:

- $x' = Ax$ has a unique solution for any initial condition $(x_0, y_0) \in \mathbb{R}^2$
- $(0, 0)$ is the only equilibrium solution of $x' = Ax$

- If $a = 0$, the solutions are periodic and will form elliptic orbits around $(0, 0)$
- If $a > 0$, the solution curves will be elliptic spirals *away* from $(0, 0)$
- If $a < 0$, the solution curves will be elliptic spirals *toward* from $(0, 0)$
- The direction of the spiral (clockwise vs. counterclockwise) can be determined by checking a single test point.

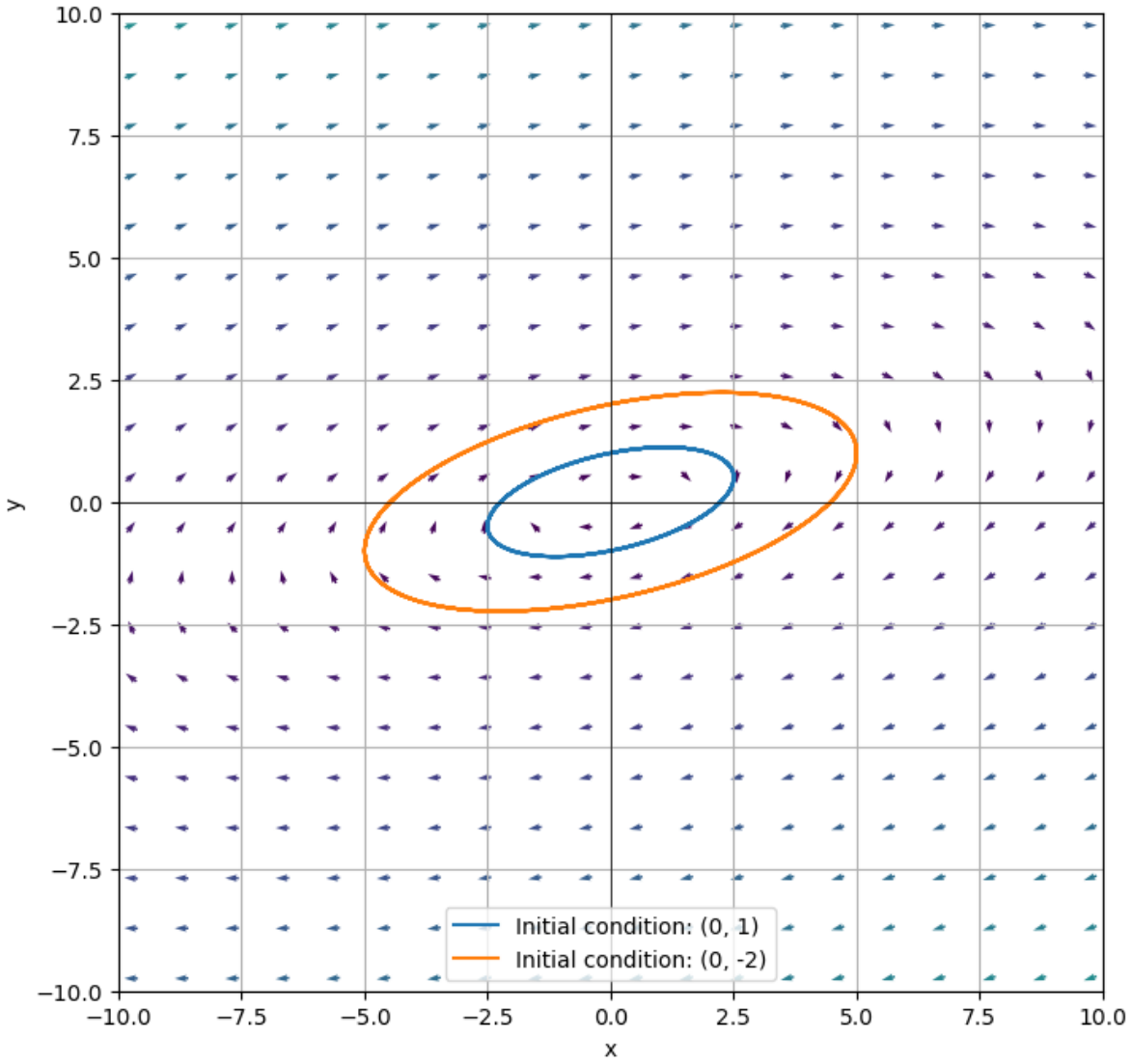
In general, we care about the correct behavior and the correct spin. We do not need additional detail.

Example

Consider the two-dimensional homogeneous linear system

$$x' = \begin{bmatrix} -1 & 5 \\ -1 & 1 \end{bmatrix} x$$

which has eigenvalues $\lambda_1 = -2i$ and $\lambda_2 = 2i$ with respective eigenvectors $v_1 = \begin{bmatrix} 1 - 2i \\ 1 \end{bmatrix}$ and $v_2 = \begin{bmatrix} 1 + 2i \\ 1 \end{bmatrix}$.

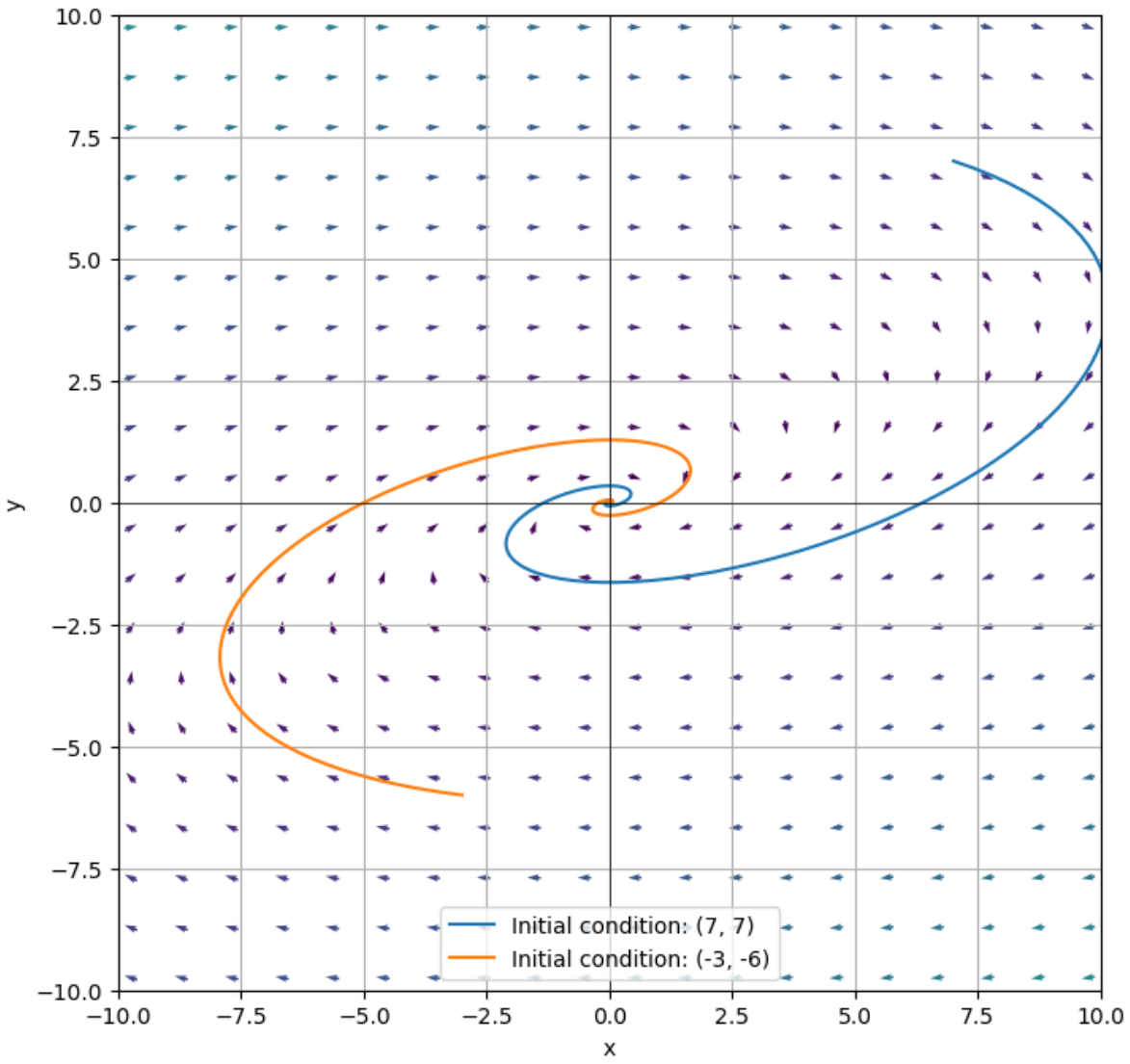


Example

Consider the two-dimensional homogeneous linear system

$$x' = \begin{bmatrix} -2 & 5 \\ -1 & 0 \end{bmatrix} x$$

which has eigenvalues $\lambda_1 = -1 - 2i$ and $\lambda_2 = -1 + 2i$ with respective eigenvectors $v_1 = \begin{bmatrix} 1 + 2i \\ 1 \end{bmatrix}$ and $v_2 = \begin{bmatrix} 1 - 2i \\ 1 \end{bmatrix}$.



Notice in this case, $\text{Re}(\lambda) = -1 < 0$, which tells us the spiral will move towards $(0, 0)$. We call this a spiral stable equilibrium. If $\text{Re}(\lambda) > 0$, the spiral will move away from $(0, 0)$ and is called a spiral unstable equilibrium (See Exercise 10).

11.5.6 The Phase Plane for Nonlinear Autonomous Systems

The goal of this section is to understand the behavior of nonlinear systems of differential equations by sketching the phase plane.

Given an autonomous system

$$x' = f(x, y)$$

$$y' = g(x, y)$$

the **nullclines** for x' are the set of all the curves where $x' = 0$, which can be found by solving $f(x, y) = 0$ explicitly (if possible). Similarly, the **nullclines** for y' are the set of all curves where $y' = 0$. Note that on x' nullclines, tangent vectors are vertical and on y' nullclines, tangent vectors are horizontal.

In addition, these nullclines encode the equilibrium solutions. These occur at the intersection of the x' and y' nullclines. Moreover, the arrows on the nullclines can only switch direction at an equilibrium point, that is, the intersection points, listed above.

Example

Find the nullclines for the system

$$x' = x(2 - x) - 3xy$$

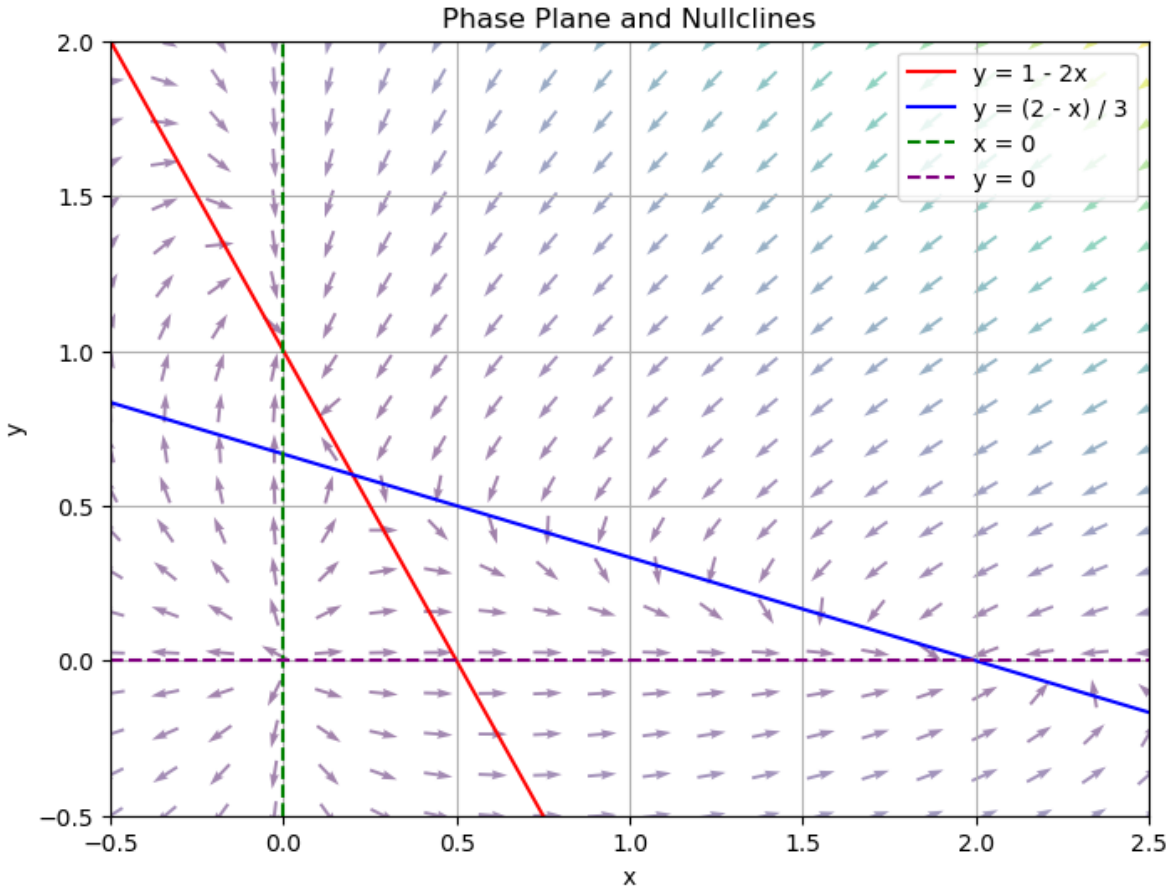
$$y' = y(1 - y) - 2xy.$$

To find the nullclines, we set each equation equal to zero.

When $x' = 0$: $x(2 - x - 3y) = 0 \to x = 0$ or $y = \dfrac{2 - x}{3}$. These are the x' nullclines.

When $y' = 0$: $y(1 - y - 2x) = 0 \to y = 0$ or $y = 1 - 2x$. These are the y' nullclines.

Notice that from these nullclines, we can also find the equilibrium solutions, by figuring out the intersection points. Notice that $(0, 0)$ is an equilibrium solution corresponding to $x = 0$, $y = 0$. $(0, 1)$ is an equilibrium solution corresponding to $x = 0$, $y = 1 - 2x$. $(2, 0)$ is an equilibrium solution corresponding to $y = \dfrac{2 - x}{3}$, $y = 0$. The last equilibrium solution is $\left(\dfrac{1}{5}, \dfrac{3}{5}\right)$, which corresponds to $y = \dfrac{2 - x}{3}$, $y = 1 - 2x$.



Given an autonomous system

$$x' = f(x, y)$$

$$y' = g(x, y)$$

with equilibrium point (a, b), the **linearization** of the system at (a, b) is the linear system

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x}\Big|_{(a,b)} & \frac{\partial f}{\partial y}\Big|_{(a,b)} \\ \frac{\partial g}{\partial x}\Big|_{(a,b)} & \frac{\partial g}{\partial y}\Big|_{(a,b)} \end{bmatrix} \begin{bmatrix} x - a \\ y - b \end{bmatrix}.$$

In addition,

$$\begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{bmatrix}$$

is called the **Jacobian Matrix** of the system and is typically denoted $J(x, y)$. If we want to evaluate it at a point, (a,b), as is done in the linearization definition, we denote that $J(a, b)$.

Example (cont.)

Find the linearization of the system

$$x' = x(2-x) - 3xy$$

$$y' = y(1-y) - 2xy$$

at each of its equilibrium solutions.

Recall from the work done previously that the equilibrium solutions are $(0,0)$, $(0,1)$, $(2,0)$, and $\left(\frac{1}{5}, \frac{3}{5}\right)$.

Now we want to find the Jacobian Matrix, $J(x,y)$.

$$J(x,y) = \begin{bmatrix} \frac{\partial}{\partial x}(2x - x^2 - 3xy) & \frac{\partial}{\partial y}(2x - x^2 - 3xy) \\ \frac{\partial}{\partial x}(y - y^2 - 2xy) & \frac{\partial}{\partial y}(y - y^2 - 2xy) \end{bmatrix} = \begin{bmatrix} 2 - 2x - 3y & -3x \\ -2y & 1 - 2y - 2x \end{bmatrix}$$

So the linearization of $(0,0)$ is

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = J(0,0) \begin{bmatrix} x - 0 \\ y - 0 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

So the linearization of $(0,1)$ is

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = J(0,1) \begin{bmatrix} x - 0 \\ y - 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ -2 & -1 \end{bmatrix} \begin{bmatrix} x \\ y - 1 \end{bmatrix}$$

So the linearization of $(2,0)$ is

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = J(2,0) \begin{bmatrix} x - 2 \\ y - 0 \end{bmatrix} = \begin{bmatrix} -2 & -6 \\ 0 & -3 \end{bmatrix} \begin{bmatrix} x - 2 \\ y \end{bmatrix}$$

So the linearization of $\left(\frac{1}{5}, \frac{3}{5}\right)$ is

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = J\left(\frac{1}{5}, \frac{3}{5}\right) \begin{bmatrix} x - \frac{1}{5} \\ y - \frac{3}{5} \end{bmatrix} = \begin{bmatrix} -\frac{1}{5} & -\frac{3}{5} \\ -\frac{6}{5} & -\frac{3}{5} \end{bmatrix} \begin{bmatrix} x - \frac{1}{5} \\ y - \frac{3}{5} \end{bmatrix}$$

Consider the linearization of $(2,0)$ found above

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -2 & -6 \\ 0 & -3 \end{bmatrix} \begin{bmatrix} x - 2 \\ y \end{bmatrix}.$$

Our hope is that the behavior of a nonlinear system is similar locally to the behavior of the linear system. In this case the eigenvalues of

$$\begin{bmatrix} -2 & -6 \\ 0 & -3 \end{bmatrix}$$

are $\lambda = -2$ and $\lambda = -3$. So we hope that locally our system behaves like a sink near $(2,0)$. Looking back at the nullclines phase plane, we do see that it behaves like a sink near $(2,0)$. It turns out that most of the time linearization describes local behavior at an equilibrium, but not always. The next result by Hartman and Grobman discusses when we can expect the linearization to describe local behavior.

Theorem (Hartman-Grobman Theorem)

Given a nonlinear autonomous system

$$x' = f(x, y)$$

$$y' = g(x, y)$$

with equilibrium point (a, b), suppose that for any eigenvalue λ of $J(a, b)$ we have $\text{Re}(\lambda) \neq 0$. Then the linearization

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = J(a, b) \begin{bmatrix} x - a \\ y - b \end{bmatrix}$$

describes the qualitative behavior of the system near the equilibrium point (a, b). This means that if $\text{Re}(\lambda) = 0$, then linearization cannot be used to describe the local behavior.

11.5.7 Exercises

Exercises

1. Which of the following sets are linearly independent?

- $\left\{ \begin{bmatrix} e^{2t} \\ e^{4t} \end{bmatrix}, \begin{bmatrix} e^{2t} \\ e^{7t} \end{bmatrix} \right\}$
- $\left\{ \begin{bmatrix} \cos(t) \\ -e^{3t} \end{bmatrix}, \begin{bmatrix} -\cos(t) \\ e^{3t} \end{bmatrix} \right\}$
- $\left\{ v_1(t) = \begin{bmatrix} \sin^2(4t) \\ t \end{bmatrix}, v_2(t) = \begin{bmatrix} \cos^2(4t) \\ 3t \end{bmatrix}, v_3(t) = \begin{bmatrix} 1 \\ 4t \end{bmatrix} \right\}$

2. Solve the initial-value problem $x' = \begin{bmatrix} -3 & 1 & -1 \\ 0 & -3 & 2 \\ 0 & -1 & 0 \end{bmatrix} x, \quad x(0) = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$.

3. Solve the initial-value problem $x' = \begin{bmatrix} 4 & -1 & 3 \\ 0 & 4 & -2 \\ 0 & 0 & 2 \end{bmatrix} x, \quad x(0) = \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}$.

4. Find the general solution to the linear system of differential equations $x' = \begin{bmatrix} -1 & 0 & 10 \\ 0 & 2 & 1 \\ 0 & -1 & 2 \end{bmatrix} x$.

5. Solve the initial-value problem $x' = \begin{bmatrix} 2 & 0 & -3 \\ 1 & 1 & -1 \\ 0 & 0 & -1 \end{bmatrix} x + \begin{bmatrix} 3e^{-2t} \\ 6e^{-2t} \\ 3e^{-2t} \end{bmatrix}, \quad x(0) = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}$.

6. Which of the following systems are autonomous?

- $\begin{cases} x' = x^2 y + y - x \\ y' = xt + yt - xy \end{cases}$.
- $\begin{cases} x' = y - x \\ y' = x + y \end{cases}$.

7. When are homogeneous linear systems with constant coefficients autonomous? That is, when is $x' = Ax$ autonomous?
8. When are nonhomogeneous linear systems with constant coefficients autonomous? That is, when is $x' = Ax+b$ autonomous?
9. Plot a phase plane for the system $x' = x(7-2x)-4xy,\ y' = -y+2xy$.
10. Consider the two-dimensional homogeneous linear system $x' = \begin{bmatrix} 0 & 10 \\ -1 & 2 \end{bmatrix} x$. Find eigenvalues and their respective eigenvectors and use this information to plot a phase plane with two sample solution curves.
11. Consider the nonlinear autonomous system $x' = x(5-x)-4xy,\ y' = y(3-y)-2xy$. Find the nullclines and equilibrium points and plot a phase plane. Then find the linearization at each equilibrium solution, and if possible, use that linearization to describe the qualitative behavior near each equilibrium solution.

11.6 Laplace Transforms

Note

In the realm of differential equations, the Laplace Transform is a crucial and powerful tool that simplifies the process of solving complex equations. This mathematical operation is named after Pierre-Simon Laplace, a French mathematician and astronomer whose work was foundational to many branches of mathematics.

The essence of the Laplace Transform lies in its ability to convert differential equations, which involve rates of change and are dependent on time, into algebraic equations, which are easier to manipulate. In this conversion process, complex patterns of time-dependent behavior are mapped into a domain where they are described as simple functions of a complex variable.

In this chapter, we will explore the fundamentals of Laplace Transforms, beginning with its definition and properties. We'll investigate how to apply Laplace Transforms to linear ordinary differential equations, including initial and boundary value problems. We will also study the Inverse Laplace Transform, a crucial technique for reclaiming our original function after performing operations in the Laplace domain. By leveraging Laplace Transforms, we can tackle systems that would be otherwise intimidating or intractable, such as those involving discontinuous or impulsive forces. This method is extensively used in various fields, including engineering, physics, and mathematics, to solve real-world problems ranging from electric circuit analysis to modeling population growth or decay.

11.6.1 Introduction to Laplace Transforms

Let $f(t)$ be a function and consider the integral

$$\mathcal{L}(f(t)) = \int_0^\infty e^{-st} f(t) \mathrm{d}t \equiv F(s).$$

If this integral is finite, then $\mathcal{L}(f(t))$ (also denoted $F(s)$) is called the **Laplace transform** of $f(t)$. Note that not all functions have Laplace transforms.

Example

Find the Laplace transform of $f(t) = 1$.

$$\mathcal{L}(1) = \int_0^\infty e^{-st} \mathrm{d}t$$

$$= \lim_{R\to\infty} \int_0^R e^{-st} \mathrm{d}t$$

$$= \lim_{R\to\infty} -\frac{1}{s} e^{-st} \Big|_{t=0}^{t=R}$$

$$= \left[\lim_{R\to\infty} -\frac{1}{s} e^{-sR}\right] + \frac{1}{s}$$

Notice that if $s < 0$, then the limit goes to ∞ and if $s > 0$, the limit goes to 0. Thus $\mathcal{L}(1) = \frac{1}{s}$ for $s > 0$.

```
The Laplace transform of f(t) = 1 is: (1/s, 0, True)
```

Example

Find the Laplace transform of $f(t) = e^{at}$.

$$\mathcal{L}(e^{at}) = \int_0^\infty e^{-st} e^{at} \mathrm{d}t$$

$$= \lim_{R\to\infty} \int_0^R e^{(a-s)t} \mathrm{d}t$$

$$= \lim_{R\to\infty} \frac{1}{a-s} e^{(a-s)t} \Big|_{t=0}^{t=R}$$

$$= \left[\lim_{R\to\infty} \frac{1}{a-s} e^{(a-s)R}\right] - \frac{1}{a-s}$$

Notice that if $s < a$, then the limit goes to ∞ and if $s > a$, the limit goes to 0. Thus $\mathcal{L}(e^{at}) = -\frac{1}{a-s} = \frac{1}{s-a}$ for $s > a$.

```
The Laplace transform of f(t) = e^(at) is: (1/(-a + s), a, True)
```

Given that a function does not always have a Laplace transform (see Exercise 4), it is worthwhile to discuss requirements for a function to have a Laplace transform. First, the function must be piecewise continuous. In other words, $f(t)$ is continuous except at possibly finitely many points. If $f(t)$ is discontinuous at t_k, then $\lim_{t\to t_k^-} f(t)$ and $\lim_{t\to t_k^+} f(t)$ are both finite. Secondly, the function must have exponential order. In other words, there must be some numbers M, a, T such that $|f(t)| \leq Me^{at}$ for $t > T$.

Linearity of Laplace Transforms: If $f(t)$ and $g(t)$ are both piecewise continuous functions of exponential order, then

$$\mathcal{L}(af(t) + bg(t)) = a\mathcal{L}(f(t)) + b\mathcal{L}(g(t)).$$

Summary

Initial Function	Laplace Transform of that Function
1	$\frac{1}{s}$, if $s > 0$
e^{at}	$\frac{1}{s-a}$, if $s > a$
$\sin(bt)$	$\frac{b}{s^2+b^2}$
$\cos(bt)$	$\frac{s}{s^2+b^2}$
$af(t)+bg(t)$	$aF(s)+bG(s)$ (Linearity)
$f'(t)$	$sF(s)-f(0)$
$f''(t)$	$s^2F(s)-sf(0)-f'(0)$

11.6.2 The Inverse Laplace Transform

Example

Consider the linear initial value problem

$$x' + 4x = e^{-5t}, \quad x(0) = 1.$$

Find the Laplace transform of the solution $x(t)$.

$$\mathcal{L}(x' + 4x) = \mathcal{L}(e^{-5t})$$

$$\mathcal{L}(x') + 4\mathcal{L}(x) = \mathcal{L}(e^{-5t})$$

$$s\mathcal{L}(x) - x(0) + 4\mathcal{L}(x) = \frac{1}{s+5}$$

$$(s+4)\mathcal{L}(x) - 1 = \frac{1}{s+5}$$

$$(s+4)\mathcal{L}(x) = \frac{1}{s+5} + \frac{s+5}{s+5}$$

$$(s+4)\mathcal{L}(x) = \frac{s+6}{s+5}$$

$$\mathcal{L}(x) = \frac{s+6}{(s+5)(s+4)}$$

Now we want to know, what function $x(t)$ has $\mathcal{L}(x) = \frac{s+6}{(s+5)(s+4)}$?

The function $f(t)$ which has Laplace transform $F(s)$ is called the **inverse Laplace transform of** $F(s)$. So long as all of the functions are piecewise continuous of exponential order, this transformation is unique, except possibly at the points of discontinuity. Moreover, the inverse Laplace transform is a linear operator. That is,

$$\mathcal{L}^{-1}(aF(s) + bG(s)) = a\mathcal{L}^{-1}(F(s)) + b\mathcal{L}^{-1}(G(s)).$$

Example (continued)

Find the inverse Laplace transform for $\mathcal{L}(x) = \dfrac{s+6}{(s+5)(s+4)}$.

First, we can use partial fractions to split this into more reasonable pieces.

$$\frac{s+6}{(s+5)(s+4)} = \frac{A}{s+5} + \frac{B}{s+4}$$

$$s+6 = A(s+4) + B(s+5)$$

If $s = -4$, then $2 = A(0) + B(1)$, thus $B = 2$. If $s = -5$, then $1 = A(-1) + B(0)$, thus $A = -1$.

So $\mathcal{L}(x) = \dfrac{-1}{s+5} + \dfrac{2}{s+4}$. Recall that $\mathcal{L}(e^{at}) = \dfrac{1}{s-a}$. Thus $\mathcal{L}(x) = -\mathcal{L}(e^{-5t}) + 2\mathcal{L}(e^{-4t})$. So $x(t) = -e^{-5t} + 2e^{-4t}$, which is the solution to the initial-value problem given in the previous example!

In order to solve differential equations with Laplace transforms, we will:

Step 1: Take the Laplace transform of both sides and set them equal.

Step 2: Enter initial conditions (if needed)

Step 3: Solve the equation algebraically for the transform $\mathcal{L}(X)$, which we will now denote $X(s)$, of the unknown function $x(t)$.

Step 4: Invert $X(s)$ to find $x(t)$.

Example

Consider the linear initial value problem

$$x'' + 4x = 8, \quad x(0) = 0, \quad x'(0) = 0.$$

Find the Laplace transform of the solution $x(t)$.

$$\mathcal{L}(x'' + 4x) = \mathcal{L}(8)$$

$$\mathcal{L}(x'') + 4\mathcal{L}(x) = \mathcal{L}(8)$$

$$s^2X(s) - sx(0) - x'(0) + 4X(s) = \frac{8}{s}$$

$$(s^2 + 4)X(s) = \frac{8}{s}$$

$$X(s) = \frac{8}{(s)(s^2 + 4)}$$

Now use partial fractions

$$\frac{8}{(s)(s^2 + 4)} = \frac{A}{s} + \frac{Bs + C}{s^2 + 4}$$

$$8 = (s^2 + 4)A + s(Bs + C)$$

Solving for A, B, and C, yields $A = 2$, $B = -2$, and $C = 0$.

Thus $X(s) = \frac{-2s}{s^2 + 4} + \frac{2}{s} = -2\left(\frac{s}{s^2 + 2^2}\right) + 2\left(\frac{1}{s}\right)$. Therefore, $x(t) = -2\cos(2t) + 2$.

11.6.3 More Laplace Transforms & the Heaviside Function

Theorem

If $f(t)$ is any piecewise continuous function of exponential order, then

$$\mathcal{L}(e^{at}f(t)) = F(s - a) \quad \text{where } F(s) = \mathcal{L}(f(t)).$$

Theorem

Theorem

If $f(t)$ is any piecewise continuous function of exponential order, then

$$\mathcal{L}(tf(t)) = -\frac{d}{ds}F(s) \quad \text{where } F(s) = \mathcal{L}(f(t)).$$

Let's update our table to take this new information into account

Initial Function	Laplace Transform
1	$\frac{1}{s}$, if $s > 0$
e^{at}	$\frac{1}{s-a}$, if $s > a$
$\sin(bt)$	$\frac{b}{s^2+b^2}$
$\cos(bt)$	$\frac{s}{s^2+b^2}$
$af(t) + bg(t)$	$aF(s) + bG(s)$ (Linearity)
$f'(t)$	$sF(s) - f(0)$
$f''(t)$	$s^2F(s) - sf(0) - f'(0)$
t	$\frac{1}{s^2}$
$tf(t)$	$-\frac{d}{ds}F(s)$
t^n	$\frac{n!}{s^{n+1}}$
te^{at}	$\frac{1}{(s-a)^2}$
$e^{at}f(t)$	$F(s-a)$
$e^{at}\sin(bt)$	$\frac{b}{(s-a)^2+b^2}$
$e^{at}\cos(bt)$	$\frac{s-a}{(s-a)^2+b^2}$

Now we will explore how we invert terms of the form $\frac{As+B}{s^2+cs+d}$ where s^2+cs+d is an irreducible quadratic. To do this, we will complete the square, that is, we will write $s^2+cs+d = s^2+cs+\left(\frac{c}{2}\right)^2+d-\left(\frac{c}{2}\right)^2 = \left(s+\frac{c}{2}\right)^2+\left(\sqrt{d-\frac{c}{2}^2}\right)^2 = (s-a)^2+b^2$. If we complete the square, then

$$\frac{As+B}{s^2+cs+d} = c_1\left(\frac{s-a}{(s-a)^2+b^2}\right)+c_2\left(\frac{b}{(s-a)^2+b^2}\right) = c_1\mathcal{L}(e^{at}\cos(bt))+c_2\mathcal{L}(e^{at}\sin(bt)).$$

Example

Find the inverse Laplace Transform of

$$\frac{4s+2}{s^2+2s+5}.$$

Notice the denominator is an irreducible quadratic, so we need to complete the square.

$$s^2+2s+5 = s^2+2s+\left(\frac{2}{2}\right)^2+5-\left(\frac{2}{2}\right)^2 = (s+1)^2+4 = (s-(-1))^2+2^2$$

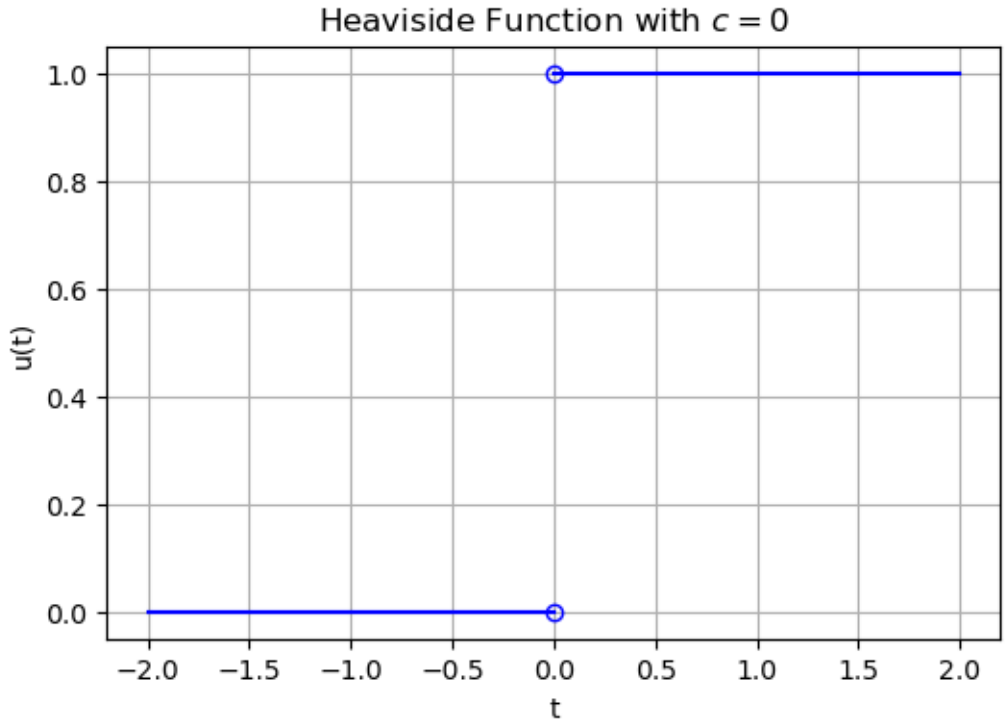
Thus,

$$\mathcal{L}^{-1}\left(\frac{4s+2}{s^2+2s+5}\right) = 4\mathcal{L}^{-1}\left(\frac{s+1}{(s+1)^2+2^2}\right)-\mathcal{L}^{-1}\left(\frac{2}{(s+1)^2+2^2}\right) = 4e^{-t}\cos(2t)-e^{-t}\sin(2t).$$

The **unit step function** (also called the **Heaviside function**) at $t = c$ is the function

$$u(t-c) = \begin{cases} 0 & \text{if } t < c \\ 1 & \text{if } t > c \end{cases}.$$

More importantly, the Laplace transform of the Heaviside function exists, and any piecewise continuous function can be expressed in terms of the Heaviside function. The Heaviside function for $c = 0$ is plotted below.

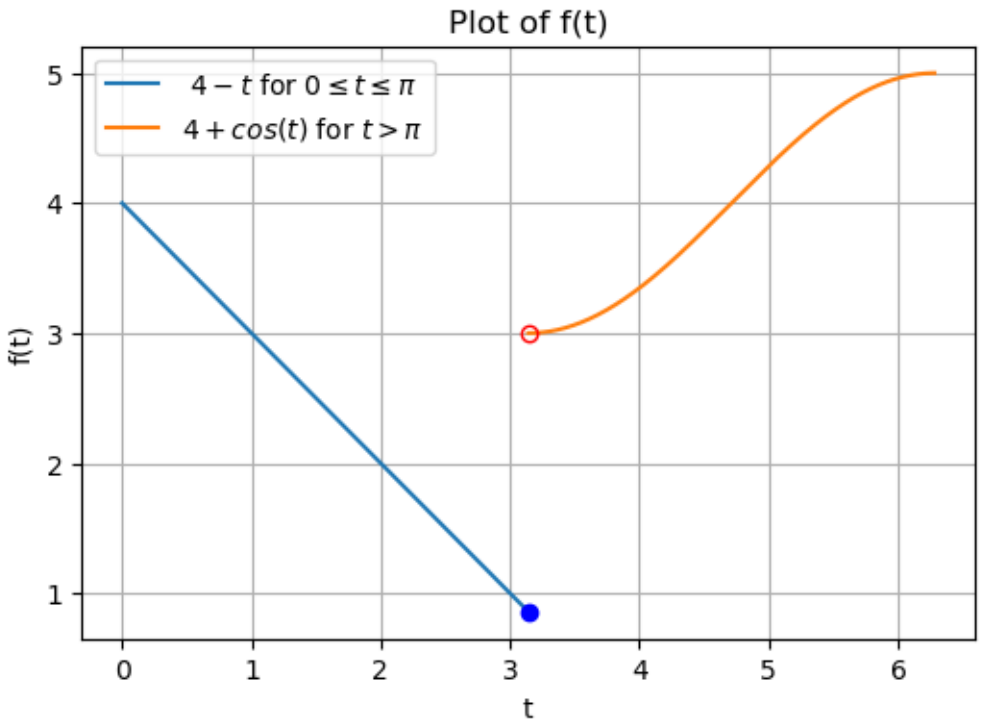


Example

Consider the function

$$f(t) = \begin{cases} 4 - t & \text{if } 0 \leq t \leq \pi \\ 4 + \cos(t) & \text{if } t > pi \end{cases}$$

plotted below. Express this function in terms of the Heaviside function.



To express $f(t)$ in terms of the Heaviside function, notice that $f(t) = 4 - t + u(t-\pi)(4+\cos(t)-(4-t))$. Now we'd like to find the Laplace transform of $u(t-\pi)(4+\cos(t)-(4-t))$ in order to be able to solve initial-value problems. The following theorem will help us do exactly that.

Theorem

Let $f(t)$ be a piecewise continuous function of exponential order and let $u(t-c)$ be the unit step function for some c. Then

- $\mathcal{L}(u(t-c)) = \dfrac{e^{-cs}}{s}$
- $\mathcal{L}(u(t-c)f(t-c)) = e^{-cs}F(s)$
- $\mathcal{L}(u(t-c)f(t)) = e^{-cs}G(s)$ where $G(s) = \mathcal{L}(f(t+c))$

Example (continued)

Consider the function

$$f(t) = \begin{cases} 4-t & \text{if } 0 \le t \le \pi \\ 4+\cos(t) & \text{if } t > \pi. \end{cases}$$

Find $\mathcal{L}(f(t))$.

Since $f(t) = 4 - t + u(t-\pi)(4+\cos(t)-(4-t))$, we know

$$\mathcal{L}(f(t)) = \mathcal{L}(4) - \mathcal{L}(t) + \mathcal{L}(u(t-\pi)(4+\cos(t)-(4-t)))$$

$$\mathcal{L}(f(t)) = \frac{4}{s} - \frac{1}{s^2} + \mathcal{L}(u(t-\pi)(4+\cos(t)-(4-t)))$$

$$\mathcal{L}(f(t)) = \frac{4}{s} - \frac{1}{s^2} + \mathcal{L}(u(t-\pi)(\cos(t)+t))$$

$$\mathcal{L}(f(t)) = \frac{4}{s} - \frac{1}{s^2} + e^{-\pi s}G(s) \text{ where } G(s) = \mathcal{L}(\cos(t)+t+\pi)$$

$$\mathcal{L}(f(t)) = \frac{4}{s} - \frac{1}{s^2} + e^{-\pi s}\left(\frac{s}{s^2+\pi^2} + \frac{1}{s^2} + \frac{\pi}{s}\right)$$

11.6.4 Exercises

Exercises

1. By hand show that the Laplace transforms of $\cos(bt)$ and $\sin(bt)$ are the functions listed in the chart.
2. Show that the Laplace transform of $f'(t)$ is $sF(s) - f(0)$.
3. Show that the Laplace transform of $f''(t)$ is $s^2F(s) - sf(0) - sf'(0)$.
4. Show that the function $g(x) = e^{t^2}$ does not have a Laplace transform.
5. Find the Laplace transform of $f(t) = 8 + 3e^{2t} - \cos(6t)$.
6. Solve the initial value problem

$$x'' + 4x' + 3x = 10\cos(t), \quad x(0) = 2, \quad x'(0) = 3.$$

7. Solve the initial value problem

$$x'' + 2x' + 3x = 9t, \quad x(0) = 1, \quad x'(0) = 2.$$

8. Let $f(t) = \begin{cases} t & \text{if } 0 \le t \le 2 \\ 0 & \text{if } t > 2 \end{cases}$. Solve the initial value problem

$$x'' + x = f(t), \quad x(0) = 0, \quad x'(0) = 0.$$

Hint: First rewrite $f(t)$ using the Heaviside function so that you can take the Laplace transform of both sides.

11.7 Solutions to Exercises

11.7.1 Section 2 Solutions

1. List the type and order of each of the following differential equations. For each ordinary differential equation, determine if it is linear or nonlinear.
   - $\frac{dQ}{dt} = \frac{tP}{100}$
     - Linear first-order ordinary differential equation
   - $\frac{\partial^2 v}{\partial t^2} + \frac{\partial^2 v}{\partial w^2} = 5$
     - Second-order partial differential equation
   - $2\frac{d^2q}{dt^2} + \cos(t)\frac{dq}{dt} = E(t)$
     - Linear second-order ordinary differential equation

2. Show that $f(x) = \sin(x) + \cos(x) + x^2 - 2$ is a solution for all x to $y'' + y = x^2$.

```
import sympy as sym

x = sym.Symbol('x')
#Define f(x)
f = sym.sin(x)+sym.cos(x)+x**2-2
print('This is the function f(x):', f)

#Define f'(x)
dfdx = sym.diff(f)
print('This is the derivative of f(x):', dfdx)

#Define f''(x)
dfdxx = sym.diff(dfdx)
print('This is the second derivative of f(x):', dfdxx)

#See if f(x) is a solution to the differential equation. If it outputs True, f(x) is␣
 ↪a solution.
dfdxx+f == x**2
```

```
This is the function f(x): x**2 + sin(x) + cos(x) - 2
This is the derivative of f(x): 2*x - sin(x) + cos(x)
This is the second derivative of f(x): -sin(x) - cos(x) + 2
```

```
True
```

Section 3 Solutions

1. Determine if the following first-order differential equations are linear, separable, both, or neither.
   - $\frac{dy}{dx} = y + x$
     - Linear, but not separable
   - $\sqrt{y} + 2\frac{dy}{dx} = e^x$
     - Neither- the square root makes this non-linear
   - $\sqrt{y}\frac{dy}{dx} = e^x$
     - Separable, but not linear
   - $\frac{dy}{dx} = x + 5$
     - Both linear and separable
2. Find the explicit solution to the differential equation $y'(x) = y + 5$.

$$\frac{dy}{y+5} = dx$$

$$\ln|y+5| = x + C$$

$$y + 5 = e^{x+C}$$

   - Solution: $y(x) = C_1 e^x - 5$

Note: The reason that the solution can be written as such is because e^{x+C} can be rewritten as $e^x e^C$; since C itself is already an arbitrary constant, e^C can simply become the new arbitrary constant, which is then named C_1. Sometimes, teachers will just name the new constant C instead of C_1 for simplicity, although C_1 is technically more accurate, as it is a new constant. This simplification is common and helps keep expressions concise.

3. Find the explicit solution to the differential equation $W'(s) = s + sW^2$.

$$\frac{dW}{1+W^2} = sds$$

$$\arctan(W) = \frac{s^2}{2} + C$$

- Solution: $W(s) = \tan\left(\frac{s^2}{2} + C\right)$

4. Find the explicit solution to the initial value problem $y'(t) = ty$ where $y(0) = 3$.

$$\frac{dy}{y} = tdt$$

$$\ln|y| = \frac{t^2}{2} + C$$

$$y = Ce^{\frac{t^2}{2}}$$

$$3 = Ce^0$$

$$C = 3$$

- Solution: $y(t) = 3e^{\frac{t^2}{2}}$

5. Find the explicit solution to the initial value problem $x'(t) = x\sin(t)$ where $x(0) = 1$.

$$\frac{dx}{x} = \sin(t)dt$$

$$\ln|x| = -\cos(t) + C$$

$$x = Ce^{-\cos(t)}$$

$$1 = Ce^{-1}$$

$$C = e$$

- Solution: $x(t) = e * e^{-\cos(t)}$

6. Show that if a first order differential equation is both linear and homogeneous, then it is also separable.

- Let $a_1(t)\frac{dx}{dt} + a_0(t)x(t) = 0$ be an arbitrary homogeneous linear first order differential equation. This means that $\frac{dx}{dt} = \frac{-a_0(t)x(t)}{a_1(t)}$. Dividing both sides by $x(t)$ allows us to separate the variables in the following way $\frac{1}{x(t)}\frac{dx}{dt} = \frac{-a_0(t)}{a_1(t)}$.

7. Find the solution to the differential equation $\frac{dP}{dt} = \frac{2P}{t} + t^2\sec^2(t)$. *Note:* This differential equation is not in standard form so we need to first put it in standard form prior to applying the method of the integrating factor.

$$\frac{dP}{dt} - \frac{2P}{t} = t^2 \sec^2(t)$$

$$\mu(t) = e^{\int \frac{-2}{t} dt} = e^{-2\ln(t)} = t^{-2}$$

$$t^{-2}\frac{dP}{dt} - \frac{2P}{t^3} = \sec^2(t)$$

$$\frac{d}{dt}(t^{-2} \cdot P(t)) = t^{-2}\frac{dP}{dt} - \frac{2P}{t^3}$$

$$\frac{d}{dt}(t^{-2} \cdot P(t)) = \sec^2(t)$$

$$\int \frac{d}{dt}(t^{-2} \cdot P(t))dt = \int \sec^2(t)dt$$

$$t^{-2} \cdot P(t) = \tan(t) + C$$

- Solution: $P(t) = t^2(\tan(t) + C)$

8. Find the solution to the initial value problem $xy' + y = \sin(x)$, where $y(\pi) = 0$. *Note:* This differential equation is not in standard form, so we need to first put it in standard form prior to applying the method of the integrating factor.

$$\frac{dy}{dx} + \frac{y}{x} = \frac{\sin(x)}{x}$$

$$\mu(t) = e^{\int \frac{1}{x} dx} = e^{\ln(x)} = x$$

$$x\frac{dy}{dx} + y = \sin(x)$$

$$\frac{d}{dx}(x \cdot y(x)) = x\frac{dy}{dx} + y$$

$$\frac{d}{dx}(x \cdot y(x)) = \sin(x)$$

$$\int \frac{d}{dx}(x \cdot y(x))dx = \int \sin(x)dx$$

$$x \cdot y(x) = -\cos(x) + C$$

$$y(x) = \frac{-\cos(x) + C}{x}$$

$$0 = \frac{-\cos(\pi) + C}{\pi}$$

$$C = -1$$

- Solution: $y(x) = \dfrac{-\cos(x) - 1}{x}$

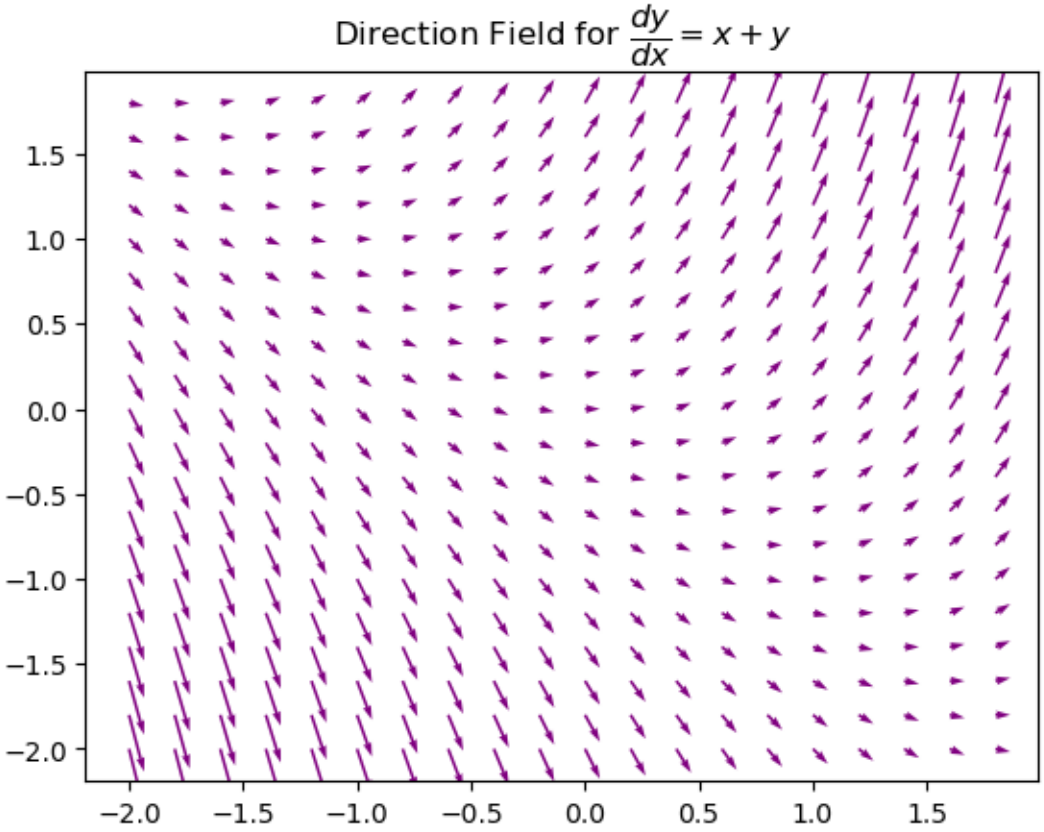
9. Plot a direction field for $y'(x) = y + x$.

```
import numpy as np
import matplotlib.pyplot as plt

#Take some time to play around with the various parameters and see how it impacts the␣
 ↪slope-field that is produced.
nx, ny = .2, .2
x = np.arange(-2,2,nx)
y = np.arange(-2,2,ny)
```

(continues on next page)

(continued from previous page)

```
#Create a rectangular grid with points
X,Y = np.meshgrid(x,y)

#Define the functions
dy = X+Y
dx = np.ones(dy.shape)

#Plot
plt.quiver(X,Y,dx,dy,color='purple')
plt.title('Direction Field for $ \dfrac{dy}{dx} = x+y $')
plt.show()
```



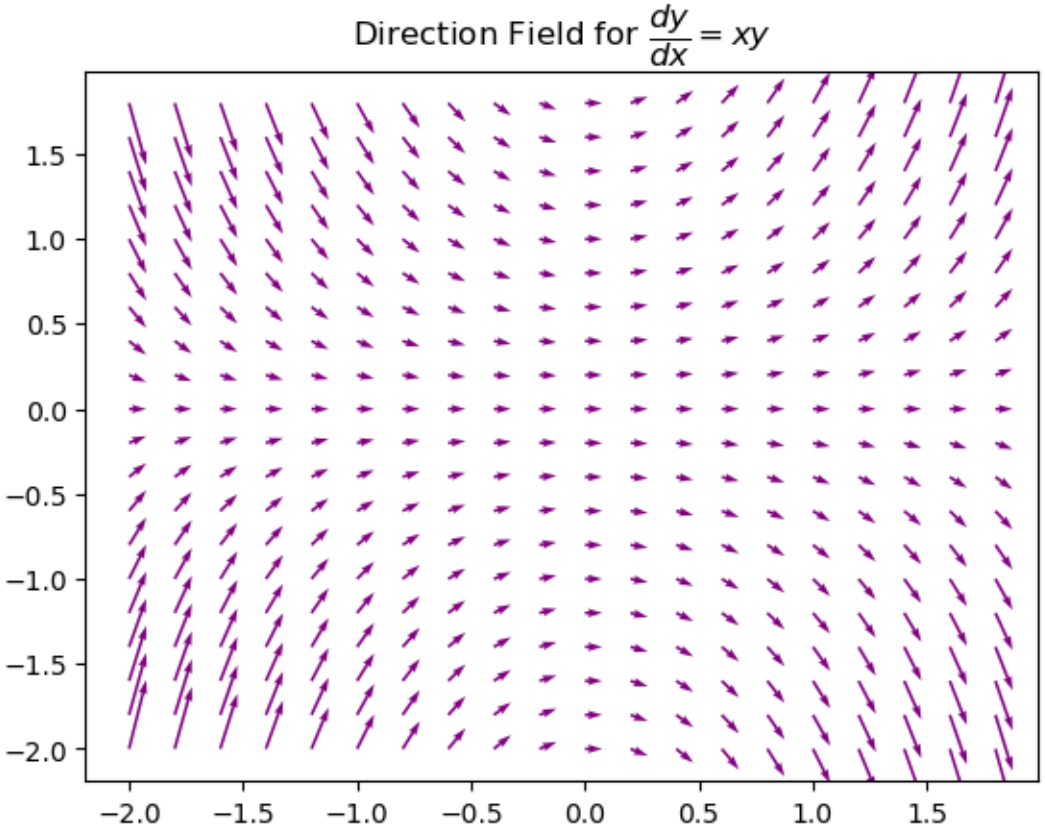
10. Plot a direction field for $y'(x) = xy$.

```
import numpy as np
import matplotlib.pyplot as plt

#Take some time to play around with the various parameters and see how it impacts the␣
↪slope-field that is produced.
nx, ny = .2, .2
x = np.arange(-2,2,nx)
y = np.arange(-2,2,ny)

#Create a rectangular grid with points
X,Y = np.meshgrid(x,y)
```

(continues on next page)

(continued from previous page)

```
#Define the functions
dy = X*Y
dx = np.ones(dy.shape)

#Plot
plt.quiver(X,Y,dx,dy,color='purple')
plt.title('Direction Field for $ \dfrac{dy}{dx} = xy $')
plt.show()
```



11. Find the equilibrium solutions of $y'(x) = \sin(y)$.

    - $\sin(y) = 0$ when $y = \dfrac{\pi k}{2}$ where k is an even integer.

12. Draw a phase line and determine where a solution of the differential equation $\frac{dT}{dt} = (T+1)T(T-3)^2$ is increasing and decreasing. Then classify the stability of each equilibrium point.

```
import numpy as np
import matplotlib.pyplot as plt

# Function for the autonomous differential equation
def f(T):
    return (T+1) * T * (T - 3)**2

# Identify the critical points (where dT/dt = 0)
critical_points = [-1, 0, 3]
```

(continues on next page)

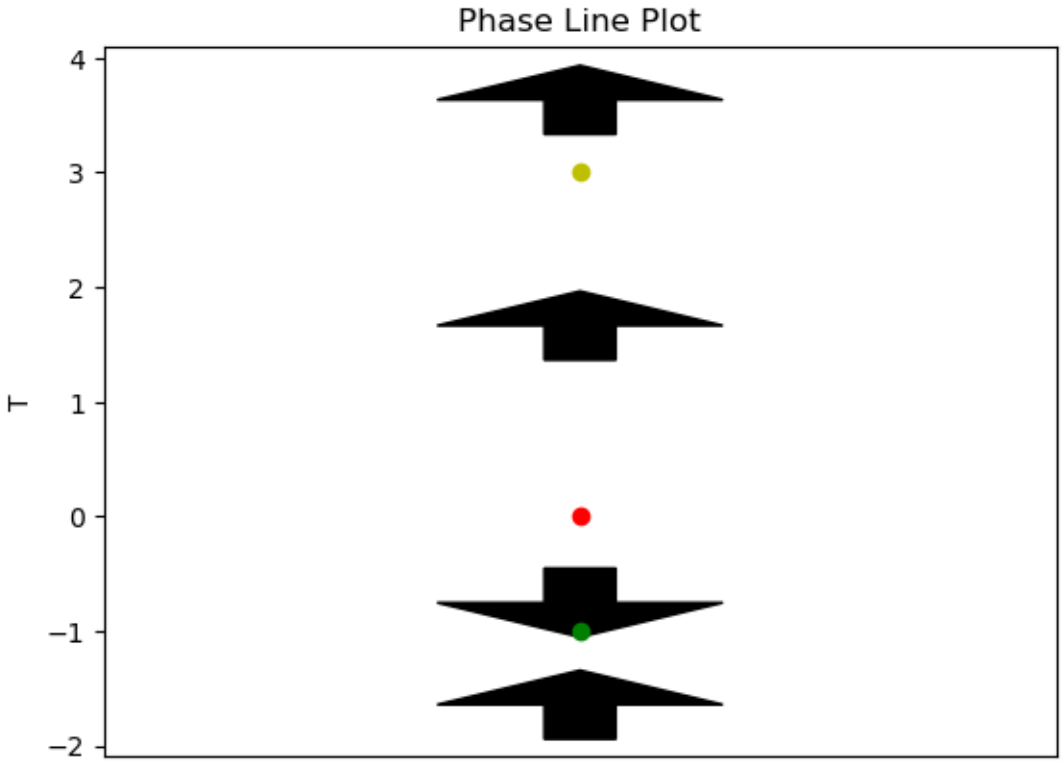
(continued from previous page)

```
# Create the figure and axis
fig, ax = plt.subplots()

# Plot the phase line at x = 0 with markers at the critical points
for point in critical_points:
    if f(point - 0.01) > 0 > f(point + 0.01):
        ax.plot(0, point, 'go')  # Stable points in green
    elif f(point - 0.01) < 0 < f(point + 0.01):
        ax.plot(0, point, 'ro')  # Unstable points in red
    else:
        ax.plot(0, point, 'yo') # Semi-stable point in yellow

# Draw arrows indicating direction of movement
arrow_properties = dict(facecolor='black', width=0.075, head_width=.3, head_length=.
 ↪3)  # Decrease head_width to make arrows skinnier

check_points = [-1.5, -0.1, 1.5, 3.5]

# Check below the lowest critical point separately
if f(check_points[0]) > 0:
    ax.arrow(0, (check_points[0] - 2)/1.8, 0, 0.3, **arrow_properties)  # Pointing up␣
 ↪if derivative is positive
else:
    ax.arrow(0, (check_points[0] - 2)/1.8, 0, -0.3, **arrow_properties)  # Pointing␣
 ↪down if derivative is negative

for i in range(len(critical_points) - 1):
    if f(check_points[i+1]) > 0:
        ax.arrow(0, (critical_points[i] + critical_points[i+1])/2.2, 0, 0.3, **arrow_
 ↪properties)  # Pointing up if derivative is positive
    else:
        ax.arrow(0, (critical_points[i] + critical_points[i+1])/2.2, 0, -0.3, **arrow_
 ↪properties)  # Pointing down if derivative is negative

# Check above the highest critical point separately
if f(check_points[-1]) > 0:
    ax.arrow(0, (critical_points[-1] + 3)/1.8, 0, 0.3, **arrow_properties)  #␣
 ↪Pointing up if derivative is positive
else:
    ax.arrow(0, (critical_points[-1] + 3)/1.8, 0, -0.3, **arrow_properties)  #␣
 ↪Pointing down if derivative is negative

# Set the y-axis limits
plt.ylim(-2.1, 4.1)
# Set the x-axis limits
plt.xlim(-0.5, 0.5)

# Label the y-axis
plt.ylabel('T')

# Title of the plot
plt.title('Phase Line Plot')

# Remove the x-axis
plt.gca().axes.get_xaxis().set_visible(False)
```

(continues on next page)

(continued from previous page)

```
# Show the plot
plt.show()
```



Using the above diagram, -1 is a stable equilibrium, 0 is an unstable equilibrium, and 3 is a semi-stable equilibrium.

Section 4 Solutions

1. Find all solutions to $x'' + x' - 6x = 0$. Suppose $x(t) = e^{kt}$ is a solution. This means it satisfies our differential equation. Since $x'(t) = ke^{kt}$ and $x''(t) = k^2e^{kt}$, we know

$x'' + x' - 6x = k^2e^{kt} + ke^{kt} - 6e^{kt} = e^{kt}(k^2 + k - 6) = 0.$

Since $e^{kt} \neq 0$, we know that $k^2 + k - 6 = 0$ when e^{kt} is a solution. Thus, $k^2 + k - 6 = 0$ is the **characteristic equation** of this differential equation. Solving the characteristic equation gives $k^2 + k - 6 = 0 = (k + 3)(k - 2) = 0$. Thus, $k = -3$ and $k = 2$ are the solutions.

This means that $\{e^{-3t}, e^{2t}\}$ *is* the fundamental set of solutions and thus, the general solution $x(t) = c_1e^{-3t} + c_2e^{2t}$ does capture *all* solutions of this differential equation.

2. Use your answer to 1 to find the general solution to the differential equation $x'' + x' - 6x = \sin(3t)$.

From Problem 1, we know that the solution to the associated homogeneous equation is $x_h(t) = c_1e^{-3t} + c_2e^{2t}$.

We now need to find the particular solution. Using the chart, we know that $x_p(t) = A\sin(3t) + B\cos(3t)$. Furthermore, $x_p'(t) = 3A\cos(3t) - 3B\sin(3t)$ and $x_p''(t) = -9A\sin(3t) - 9B\cos(3t) = -9x_p(t)$. Plugging this into the differential equation gives

$$x_p''(t) + x_p'(t) - 6x_p(t) = \sin(3t)$$

$$-9x_p(t) + x_p'(t) - 6x_p(t) = \sin(3t)$$

$$x_p'(t) - 15x_p(t) = \sin(3t)$$

$$3A\cos(3t) - 3B\sin(3t) - 15(A\sin(3t) + B\cos(3t)) = \sin(3t)$$

$$(3A - 15B)\cos(3t) + (-15A - 3B)\sin(3t) = \sin(3t)$$

Thus we know that $3A - 15B = 0$ and $-15A - 3B = 1$. This gives that $A = \frac{-5}{78}$ and $B = \frac{-1}{78}$. Thus $x_p(t) = \frac{-5}{78}\sin(3t) + \frac{-1}{78}\cos(3t)$. So the general solution is

$$x(t) = x_h(t) + x_p(t) = c_1e^{-3t} + c_2e^{2t} + \frac{-5}{78}\sin(3t) + \frac{-1}{78}\cos(3t).$$

3. Use your answer to 1 to find the general solution to the differential equation $x'' + x' - 6x = 5e^{4t}$.

From Problem 1, we know that the solution to the associated homogeneous equation is $x_h(t) = c_1e^{-3t} + c_2e^{2t}$.

We now need to find the particular solution. Using the chart, we know that $x_p(t) = Ae^{4t}$. Furthermore, $x_p'(t) = 4Ae^{4t}$ and $x_p''(t) = 16Ae^{4t}$. Plugging this into the differential equation gives

$$x_p''(t) + x_p'(t) - 6x_p(t) = 5e^{4t}$$

$$16Ae^{4t} + 4Ae^{4t} - 6Ae^{4t} = 5e^{4t}$$

$$14Ae^{4t} = 5e^{4t}$$

Thus we know $A = \frac{5}{14}$. Thus $x_p(t) = \frac{5}{14}e^{4t}$. So the general solution is

$$x(t) = x_h(t) + x_p(t) = c_1e^{-3t} + c_2e^{2t} + \frac{5}{14}e^{4t}.$$

4. Use your answer to 1 to find the general solution to the differential equation $x'' + x' - 6x = 5e^{2t}$. Hint: Think about why changing the exponent from $4t$ to $2t$ adds an extra step to the problem.

From Problem 1, we know that the solution to the associated homogeneous equation is $x_h(t) = c_1e^{-3t} + c_2e^{2t}$.

We now need to find the particular solution. Using the chart, we know that $x_p(t) = Ae^{2t}$. But, this isn't linearly independent from $x_h(t)$ so we must multiple our particular solution by t. Therefore, our updated $x_p(t) = Ate^{2t}$. Furthermore, $x_p'(t) = Ae^{2t} + 2tAe^{2t}$ and $x_p''(t) = 4Ae^{2t} + 4Ate^{2t}$. Plugging this into the differential equation gives

$$x_p''(t) + x_p'(t) - 6x_p(t) = 5e^{2t}$$

$$4Ae^{2t} + 4Ate^{2t} + Ae^{2t} + 2tAe^{2t} - 6Ate^{2t} = 5e^{2t}$$

$$5Ae^{2t} = 5e^{2t}$$

Thus we know $A = 1$. Thus $x_p(t) = te^{2t}$. So the general solution is

$$x(t) = x_h(t) + x_p(t) = c_1e^{-3t} + c_2e^{2t} + te^{2t}.$$

5. Use your answer to 2 and 3 to find the general solution of the differential equation $x'' + x' - 6x = 5e^{4t} + \sin(3t)$. Hint: What does the Superposition Principle tell us?

From Problem 2 and Problem 3, we can read off the general solution. In particular, we know

$$x(t) = c_1e^{-3t} + c_2e^{2t} + \frac{5}{14}e^{4t} + \frac{-5}{78}\sin(3t) + \frac{-1}{78}\cos(3t).$$

6. Find the form of the particular solution of the following differential equations. Note you do not need to solve for the undetermined coefficients.

- $x'' - 6x' + 9x = 3t + 10\cos(t)$
  - $x_p(t) = At + B + C\cos(t) + D\sin(t)$
- $x'' - 9x = te^{3t}$
  - $x_p(t) = (At + B)Ce^{3t}$, which can be simplified to $x_p(t) = (Dt + E)e^{3t}$
- $2x'' + x' + 4x = 3t^2\sin(2t)$
  - $x_p(t) = (At^2 + Bt + C)(D\cos(2t) + E\sin(2t))$

Section 5 Solutions

1. Which of the following sets are linearly independent?

- $\left\{\begin{bmatrix} e^{2t} \\ e^{4t} \end{bmatrix}, \begin{bmatrix} e^{2t} \\ e^{7t} \end{bmatrix}\right\}$
  - These are linearly independent because of the second coordinate in each vector.
- $\left\{\begin{bmatrix} \cos(t) \\ -e^{3t} \end{bmatrix}, \begin{bmatrix} -\cos(t) \\ e^{3t} \end{bmatrix}\right\}$
  - These are not linearly independent. Notice that adding the two vectors together produces the zero vector.
- $\left\{v_1(t) = \begin{bmatrix} \sin^2(4t) \\ t \end{bmatrix}, v_2(t) = \begin{bmatrix} \cos^2(4t) \\ 3t \end{bmatrix}, v_3(t) = \begin{bmatrix} 1 \\ 4t \end{bmatrix}\right\}$
  - These are not linearly independent as $v_1(t) + v_2(t) - v_3(t) = \vec{0}$

2. Solve the initial-value problem

$$x' = \begin{bmatrix} -3 & 1 & -1 \\ 0 & -3 & 2 \\ 0 & -1 & 0 \end{bmatrix} x, \quad x(0) = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}.$$

$$\det(A - \lambda I) = 0$$

$$\det\left(\begin{bmatrix} -3-\lambda & 1 & -1 \\ 0 & -3-\lambda & 2 \\ 0 & -1 & -\lambda \end{bmatrix}\right) = 0$$

$$(-3-\lambda)(\lambda^2 + 3\lambda + 2) = 0$$

$$(-3-\lambda)(\lambda+1)(\lambda+2) = 0$$

Thus the eigenvalues are $\lambda = -3$, $\lambda = -2$, and $\lambda = -1$.

Now we want to find the eigenvectors associated to each eigenvalue. Let's start with $\lambda = -3$.

Setting up the augmented matrix yields

$$\left[\begin{array}{ccc|c} \begin{bmatrix} 0 & 1 & -1 \\ 0 & 0 & 2 \\ 0 & -1 & -3 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \end{array}\right] \approx \left[\begin{array}{ccc|c} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \end{array}\right]$$

Thus an eigenvector associated to $\lambda = -3$ is $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$.

Let's now consider $\lambda = -2$.

Setting up the augmented matrix yields

$$\left[\begin{array}{ccc|c} -1 & 1 & -1 & 0 \\ 0 & -1 & 2 & 0 \\ 0 & -1 & -2 & 0 \end{array}\right] \approx \left[\begin{array}{ccc|c} 1 & 0 & -1 & 0 \\ 0 & 1 & -2 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

Thus an eigenvector associated to $\lambda = -2$ is $\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$.

Let's now consider $\lambda = -1$.

Setting up the augmented matrix yields

$$\left[\begin{array}{ccc|c} -2 & 1 & -1 & 0 \\ 0 & -2 & 2 & 0 \\ 0 & -1 & 1 & 0 \end{array}\right] \approx \left[\begin{array}{ccc|c} -2 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

Thus an eigenvector associated to $\lambda = -1$ is $\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$.

This means the general solution to this system is $x(t) = c_1 \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} e^{-t} + c_2 \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} e^{-2t} + c_3 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} e^{-3t}$.

Using the initial condition, we find that

$$\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = c_1 \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} e^{-(0)} + c_2 \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} e^{-2(0)} + c_3 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} e^{-3(0)}$$

$$\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = c_1 \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} + c_2 \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} + c_3 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Solving this system for c_1, c_2, and c_3 gives $c_1 = -1$, $c_2 = 1$, and $c_3 = 0$.

Thus the solution to the initial value problem is

$$x(t) = -\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} e^{-t} + \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} e^{-2t}.$$

3. Solve the initial-value problem

$$x' = \begin{bmatrix} 4 & -1 & 3 \\ 0 & 4 & -2 \\ 0 & 0 & 2 \end{bmatrix} x, \quad x(0) = \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}.$$

Since this is a triangular matrix, we know the eigenvalues are $\lambda = 4$, $\lambda = 4$, and $\lambda = 2$.

Now we want to find the eigenvectors associated to each eigenvalue. Let's start with $\lambda = 4$.

Setting up the augmented matrix yields

$$\left[\begin{array}{ccc|c} 0 & -1 & 3 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & -2 & 0 \end{array}\right] \approx \left[\begin{array}{ccc|c} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

Thus an eigenvector associated to $\lambda = 4$ is $\begin{bmatrix}1\\0\\0\end{bmatrix}$. But notice that $\lambda = 4$ has multiplicity 2 and we only have a single eigenvector associated to it. This means the eigenvalue is defective. This means we need to now find $\vec{w}$.

$$(A - 4I)\vec{w} = \begin{bmatrix}1\\0\\0\end{bmatrix}$$

$$\left[\begin{array}{ccc|c}0 & -1 & 3 & 1\\0 & 0 & -2 & 0\\0 & 0 & -2 & 0\end{array}\right] \approx \left[\begin{array}{ccc|c}0 & 1 & 0 & -1\\0 & 0 & 1 & 0\\0 & 0 & 0 & 0\end{array}\right]$$

Thus $\vec{w} = \begin{bmatrix}0\\-1\\0\end{bmatrix} + w_1 \begin{bmatrix}1\\0\\0\end{bmatrix}$. So one possible choice for $\vec{w} = \begin{bmatrix}1\\-1\\0\end{bmatrix}$. But notice that $\lambda = 4$ has multiplicity 2 and we only have a single eigenvector associated to it. This means the eigenvalue is defective. This means we need to now find $\vec{w}$.

Furthermore, you should find that an eigenvector associated to $\lambda = 2$ is $\begin{bmatrix}-1\\1\\1\end{bmatrix}$.

Let's now consider $\lambda = -1$.

This means the general solution to this system is $x(t) = c_1 \begin{bmatrix}1\\0\\0\end{bmatrix} e^{4t} + c_2 \left(\begin{bmatrix}1\\0\\0\end{bmatrix} te^{4t} + \begin{bmatrix}1\\-1\\0\end{bmatrix} e^{4t} \right) + c_3 \begin{bmatrix}-1\\1\\1\end{bmatrix} e^{2t}$.

Using the initial condition, we find that

$$\begin{bmatrix}0\\0\\2\end{bmatrix} = c_1 \begin{bmatrix}1\\0\\0\end{bmatrix} e^{4(0)} + c_2 \left(\begin{bmatrix}1\\0\\0\end{bmatrix} (0)e^{4(0)} + \begin{bmatrix}1\\-1\\0\end{bmatrix} e^{4(0)} \right) + c_3 \begin{bmatrix}-1\\1\\1\end{bmatrix} e^{2(0)}$$

$$\begin{bmatrix}0\\0\\2\end{bmatrix} = c_1 \begin{bmatrix}1\\0\\0\end{bmatrix} + c_2 \begin{bmatrix}1\\-1\\0\end{bmatrix} + c_3 \begin{bmatrix}-1\\1\\1\end{bmatrix}$$

Solving this system for c_1, c_2, and c_3 gives $c_1 = 0$, $c_2 = 2$, and $c_3 = 2$.

Thus the solution to the initial value problem is $x(t) = 2 \left(\begin{bmatrix}1\\0\\0\end{bmatrix} te^{4t} + \begin{bmatrix}1\\-1\\0\end{bmatrix} e^{4t} \right) + 2 \begin{bmatrix}-1\\1\\1\end{bmatrix} e^{2t}$.

4. Find the general solution to the linear system of differential equations

$$x' = \begin{bmatrix}-1 & 0 & 10\\0 & 2 & 1\\0 & -1 & 2\end{bmatrix} x.$$

$$\det(A - \lambda I) = 0$$

$$(-1 - \lambda)(\lambda^2 - 4\lambda + 5) = 0$$

Thus the eigenvalues are $\lambda = -1$ and $\lambda = 2 \pm i$.

You should find that an eigenvector associated to $\lambda = -1$ is $\begin{bmatrix}1\\0\\0\end{bmatrix}$.

Now we want to find the eigenvector associated with $\lambda = 2 + i$.

Setting up the augmented matrix yields

$$\left[\begin{array}{ccc|c} -3-i & 0 & 10 & 0 \\ 0 & -i & 1 & 0 \\ 0 & -1 & -i & 0 \end{array}\right] \approx \left[\begin{array}{ccc|c} 1 & 0 & -3+i & 0 \\ 0 & 1 & i & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

Thus the eigenvector associated to $\lambda = 2+i$ is $\begin{bmatrix} 3-i \\ -i \\ 1 \end{bmatrix}$.

We also know that the eigenvector associated to $\lambda = 2-i$ is the complex conjugate of $\begin{bmatrix} 3-i \\ -i \\ 1 \end{bmatrix}$. Thus the eigenvector associated to $\lambda = 2-i$ is $\begin{bmatrix} 3+i \\ i \\ 1 \end{bmatrix}$.

This means that a fundamental solution set is $\left\{ \begin{bmatrix} 3-i \\ -i \\ 1 \end{bmatrix} e^{(2+i)t}, \begin{bmatrix} 3+i \\ i \\ 1 \end{bmatrix} e^{(2-i)t} \right\}$. But if we want a real-valued solution (which is better for interpreting physical situations), we need to find a different fundamental solution set.

Step 1 & Step 2: Let $x_1(t) = \begin{bmatrix} 3-i \\ -i \\ 1 \end{bmatrix} e^{(2+i)t}$.

Step 3: Using Euler's formula and algebra $x_1(t) = e^{2t} \begin{bmatrix} 3-i \\ -i \\ 1 \end{bmatrix} (\cos(t) + i\sin(2t)) = e^{2t} \begin{bmatrix} 3\cos(t) + \sin(t) \\ \sin(t) \\ \cos(t) \end{bmatrix} + ie^{2t} \begin{bmatrix} -\cos(t) + 3\sin(t) \\ -\cos(t) \\ \sin(t) \end{bmatrix}$.

This means that the general solution to this system of differential equations is

$$x(t) = c_1 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} e^{-t} + c_2 e^{2t} \begin{bmatrix} 3\cos(t) + \sin(t) \\ \sin(t) \\ \cos(t) \end{bmatrix} + c_3 e^{2t} \begin{bmatrix} -\cos(t) + 3\sin(t) \\ -\cos(t) \\ \sin(t) \end{bmatrix}.$$

5. Solve the initial-value problem

$$x' = \begin{bmatrix} 2 & 0 & -3 \\ 1 & 1 & -1 \\ 0 & 0 & -1 \end{bmatrix} x + \begin{bmatrix} 3e^{-2t} \\ 6e^{-2t} \\ 3e^{-2t} \end{bmatrix}, \quad x(0) = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}.$$

- $x' = \begin{bmatrix} 2 & 0 & -3 \\ 1 & 1 & -1 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ t \\ 2 \end{bmatrix} + \begin{bmatrix} 3e^{-2t} \\ 6e^{-2t} \\ 3e^{-2t} \end{bmatrix}$

6. Which of the following systems are autonomous?

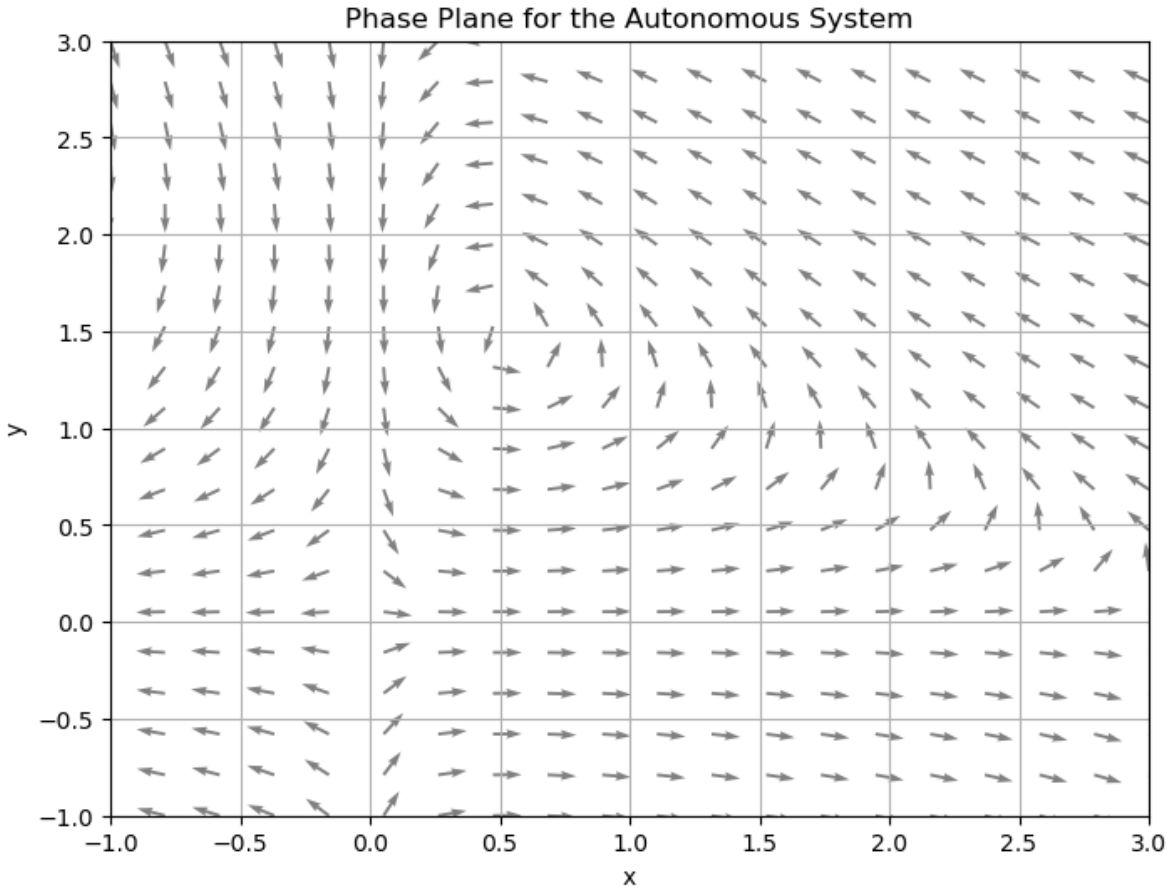
- $\begin{cases} x' = x^2 y + y - x \\ y' = xt + yt - xy \end{cases}$
  - Is not autonomous because the independent variable t appears.
- $\begin{cases} x' = y - x \\ y' = x + y \end{cases}$
  - Is autonomous

7. When are homogeneous linear systems with constant coefficients autonomous? That is, when is $x' = Ax$ autonomous?
    - These systems are always autonomous!
8. When are nonhomogeneous linear systems with constant coefficients autonomous? That is, when is $x' = Ax + b$ autonomous?
    - These are sometimes autonomous. In particular, they are autonomous when b only involves constants.
9. Plot a phase plane for the system

$$x' = x(7 - 2x) - 4xy$$

$$y' = -y + 2xy.$$

```
import numpy as np
import matplotlib.pyplot as plt

# Set the range of values for x and y
x = np.linspace(-1, 3, 20)
y = np.linspace(-1, 3, 20)

# Create a grid of points (x, y)
X, Y = np.meshgrid(x, y)

# Define the system of equations
dx = X*(7 - 2*X) - 4*X*Y
dy = -Y + 2*X*Y

# Normalize the arrows to have the same length
N = np.sqrt(dx**2 + dy**2)
dx = dx/N
dy = dy/N

# Create the quiver plot
plt.figure(figsize=(8, 6))
plt.quiver(X, Y, dx, dy, color='gray')

plt.title('Phase Plane for the Autonomous System')
plt.xlabel('x')
plt.ylabel('y')
plt.xlim([-1, 3])
plt.ylim([-1, 3])
plt.grid(True)
plt.show()
```



10. Consider the two-dimensional homogeneous linear system

$$x' = \begin{bmatrix} 0 & 10 \\ -1 & 2 \end{bmatrix} x.$$

Find eigenvalues and their respective eigenvectors and use this information to plot a phase plane with two sample solution curves.

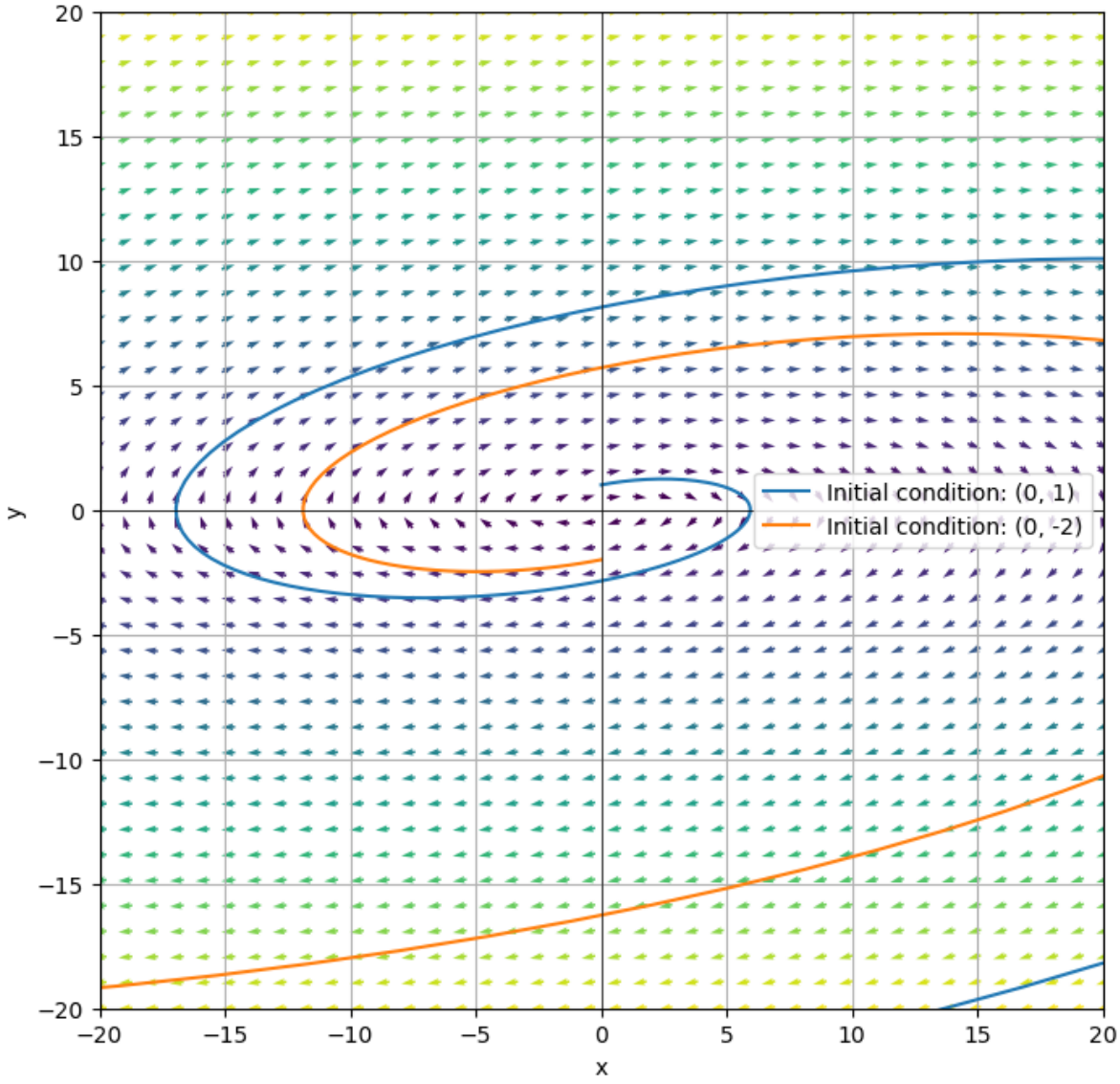
The eigenvalues of A are $\lambda = 1 \pm 3i$. The eigenvector associated to $\lambda = 1+3i$ is $\begin{bmatrix} 1-3i \\ 1 \end{bmatrix}$ and the eigenvector associated to $\lambda = 1-3i$ is $\begin{bmatrix} 1+3i \\ 1 \end{bmatrix}$.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

# Define the system of equations
def system(t, state):
    x, y = state
    dx = 10*y
    dy = -1*x + 2*y
    return [dx, dy]

# Set up the grid
Y, X = np.mgrid[-20:20:40j, -20:20:40j]
DX, DY = system(None, [X, Y])
```

(continues on next page)

(continued from previous page)

```
# Normalize vectors to a consistent length for visualization
M = (np.hypot(DX, DY))
M[M == 0] = 1.
DX /= M
DY /= M

fig, ax = plt.subplots(figsize=(8, 8))

# Plot the vector field using quiver
ax.quiver(X, Y, DX, DY, M, pivot='mid')

# Define the eigenvalues and eigenvectors
lambda1 = 1-3j
lambda2 = 1+3j
v1 = np.array([1+3j, 1])
v2 = np.array([1-3j, 1])

x = np.linspace(-10, 10, 800)

# Plot solution curves for two initial conditions
t_span = [-10, 10]
t_eval = np.linspace(t_span[0], t_span[1], 1000)

for initial_condition in [(0, 1), (0, -2)]:
    sol = solve_ivp(system, t_span, initial_condition, t_eval=t_eval)
    ax.plot(sol.y[0], sol.y[1],label=f'Initial condition: {initial_condition}')

ax.set_xlabel('x')
ax.set_ylabel('y')
ax.grid(True)
ax.axvline(0, color='black',linewidth=0.5)
ax.axhline(0, color='black',linewidth=0.5)
plt.xlim([-20, 20])
plt.ylim([-20, 20])
ax.legend()
plt.show()
```



11. Consider the nonlinear autonomous system

$$x' = x(5 - x) - 4xy$$

$$y' = y(3 - y) - 2xy.$$

Find the nullclines and equilibrium points and plot a phase plane. Then find the linearization at each equilibrium solution, and if possible, use that linearization to describe the qualitative behavior near each equilibrium solution.

To find the nullclines, we set each equation equal to zero.

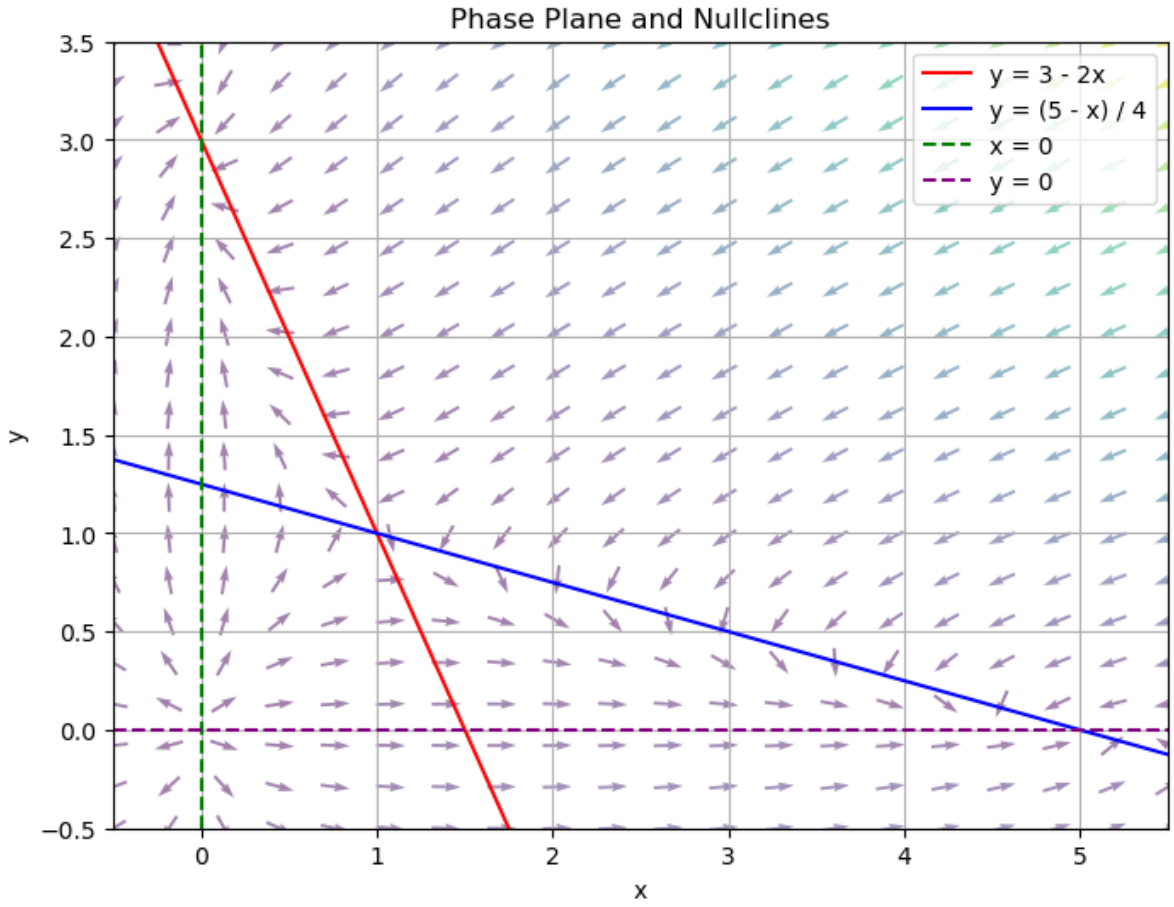
When $x' = 0$: $x(5 - x - 4y) = 0 \to x = 0$ or $y = \dfrac{5 - x}{4}$. These are the x' nullclines.

When $y' = 0$: $y(3 - y - 2x) = 0 \to y = 0$ or $y = 3 - 2x$. These are the y' nullclines.

Notice that from these nullclines, we can also find the equilibrium solutions, by figuring out the intersection points. Notice that $(0, 0)$ is an equilibrium solution corresponding to $x = 0$, $y = 0$. $(0, 3)$ is an equilibrium solution corresponding to $x = 0$, $y = 3 - 2x$. $(5, 0)$ is an equilibrium solution corresponding to $y = \dfrac{5 - x}{4}$, $y = 0$. The last equilibrium solution

is $(1, 1)$, which corresponds to $y = \dfrac{5 - x}{4}$, $y = 3 - 2x$.

```
import numpy as np
import matplotlib.pyplot as plt

# Define the system of equations
def system(x, y):
    dx = x * (5 - x) - 4 * x * y
    dy = y * (3 - y) - 2 * x * y
    return dx, dy

# Create an x, y grid
y, x = np.mgrid[-0.5:3.5:20j, -0.5:5.5:20j]

# Calculate dx, dy for the vector field
dx, dy = system(x, y)

# Normalize the arrows so their size represents their speed
M = (np.hypot(dx, dy))
M[M == 0] = 1.
dx /= M
dy /= M

plt.figure(figsize=(8, 6))

# Plot the vector field using quiver
plt.quiver(x, y, dx, dy, M, pivot='mid', alpha=0.5)

# Create an x vector
x = np.linspace(-0.5, 5.5, 400)

# Define nullclines
y1 = 3 - 2 * x
y2 = (5 - x) / 4

# Plot nullclines
plt.plot(x, y1, 'r', label="y = 3 - 2x")
plt.plot(x, y2, 'b', label="y = (5 - x) / 4")

# Also plot x=0 and y=0
plt.axvline(0, color='green', linestyle='--', label="x = 0")
plt.axhline(0, color='purple', linestyle='--', label="y = 0")

# Labels and legends
plt.xlabel('x')
plt.ylabel('y')
plt.legend()

# Set the x and y axis cutoffs
plt.ylim([-0.5,3.5])
plt.xlim([-0.5,5.5])

plt.grid(True)
plt.title('Phase Plane and Nullclines')
plt.show()
```



Now we want to find the Jacobian Matrix, $J(x, y)$.

$$J(x,y) = \begin{bmatrix} \frac{\partial}{\partial x}(5x - x^2 - 4xy) & \frac{\partial}{\partial y}(5x - x^2 - 4xy) \\ \frac{\partial}{\partial x}(3y - y^2 - 2xy) & \frac{\partial}{\partial y}(3y - y^2 - 2xy) \end{bmatrix} = \begin{bmatrix} 5 - 2x - 4y & -4x \\ -2y & 3 - 2y - 2x \end{bmatrix}$$

So the linearization of $(0, 0)$ is

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = J(0,0) \begin{bmatrix} x - 0 \\ y - 0 \end{bmatrix} = \begin{bmatrix} 5 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

So the linearization of $(0, 3)$ is

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = J(0,3) \begin{bmatrix} x - 0 \\ y - 3 \end{bmatrix} = \begin{bmatrix} -2 & 0 \\ -6 & -3 \end{bmatrix} \begin{bmatrix} x \\ y - 3 \end{bmatrix}$$

So the linearization of $(5, 0)$ is

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = J(5,0) \begin{bmatrix} x - 5 \\ y - 0 \end{bmatrix} = \begin{bmatrix} -5 & -20 \\ 0 & -7 \end{bmatrix} \begin{bmatrix} x - 5 \\ y \end{bmatrix}$$

So the linearization of $(1, 1)$ is

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = J(1,1) \begin{bmatrix} x - 1 \\ y - 1 \end{bmatrix} = \begin{bmatrix} -1 & -4 \\ -2 & -1 \end{bmatrix} \begin{bmatrix} x - 1 \\ y - 1 \end{bmatrix}$$

From this analysis, we would expect $(0, 0)$ to behave like a source. We would expect $(0, 3)$ to behave like a sink. We would expect $(5, 0)$ to behave like a sink. We would expect $(1, 1)$ to behave like a sink and a source (depending on where we approach it).

Section 6 Solutions

1. By hand show that the Laplace transforms of $\cos(bt)$ and $\sin(bt)$ are the functions listed in the chart. *Recall: Euler's formula says $e^{i(bt)} = \cos(bt) + i\sin(bt)$.

$$\mathcal{L}(e^{ibt}) = \int_0^\infty e^{-st} e^{ibt} \mathrm{d}t$$

$$\mathcal{L}(e^{ibt}) = \int_0^\infty e^{-st}(\cos(bt))\mathrm{d}t + i\int_0^\infty e^{-st}(\sin(bt))\mathrm{d}t$$

$$= \mathcal{L}(\cos(bt)) + i\mathcal{L}(\sin(bt))$$

Since

$$\mathcal{L}(e^{ibt}) = \frac{1}{s-ib}\frac{s+ib}{s+ib} = \frac{s+ib}{s^2+b^2} = \frac{s}{s^2+b^2} + i\frac{b}{s^2+b^2},$$

we know that $\mathcal{L}(\cos(bt)) = \dfrac{s}{s^2-b^2}$ and $\mathcal{L}(\sin(bt)) = \dfrac{b}{s^2-b^2}$ for $s > |ib|$.

2. Show that the Laplace transform of $f'(t)$ is $sF(s) - f(0)$.

We know

$$\mathcal{L}(f'(t)) = \int_0^\infty e^{-st} f'(t)\mathrm{d}t.$$

Applying integration by parts yields

$$\mathcal{L}(f'(t)) = \lim_{r\to\infty}(e^{-st}f(t))\Big|_{t=0}^{t=r} + s\int_0^\infty e^{-st}f(t)dt.$$

Notice that

$$s\int_0^\infty e^{-st}f(t)\mathrm{d}t = F(s), \lim_{r\to\infty}(e^{-sr}f(r)) = 0$$

because $f(t)$ is of exponential order. Thus

$$\mathcal{L}(f'(t)) = [0 - f(0)] + sF(s) = sF(s) - f(0).$$

3. Show that the Laplace transform of $f''(t)$ is $s^2F(s) - sf(0) - f'(0)$.

We know

$$\mathcal{L}(f''(t)) = \int_0^\infty e^{-st} f''(t)\mathrm{d}t.$$

Applying integration by parts yields

$$\mathcal{L}(f''(t)) = \lim_{r\to\infty}(e^{-st}f'(t))\Big|_{t=0}^{t=r} + s\int_0^\infty e^{-st}f'(t)\mathrm{d}t.$$

Notice that

$$s\int_0^\infty e^{-st}f'(t)\mathrm{d}t = s(sF(s) - f(0)), \lim_{r\to\infty}(e^{-sr}f'(r)) = 0$$

because $f'(t)$ is of exponential order. Thus

$$\mathcal{L}(f''(t)) = [0 - f'(0)] + s(sF(s) - f(0)) = s^2F(s) - sf(0) - f'(0).$$

4. Show that the function $g(x) = e^{t^2}$ does not have a Laplace transform.

$$\mathcal{L}(e^{t^2}) = \int_0^\infty e^{-st} e^{t^2} \mathrm{d}t = \infty$$

Thus, the Laplace transform is not defined.

5. Find the Laplace transform of $f(t) = 8 + 3e^{2t} - \cos(6t)$.

$$\begin{aligned}\mathcal{L}(f(t)) &= 8\mathcal{L}(1) + 3\mathcal{L}(e^{2t}) - \mathcal{L}(\cos(6t)) \\ &= \frac{8}{s} + \frac{3}{s-2} + \frac{s}{s^2+36}.\end{aligned}$$

6. Solve the initial value problem

$$x'' + 4x' + 3x = 10\cos(t), \quad x(0) = 2, \quad x'(0) = 3.$$

$$\mathcal{L}(x'' + 4x' + 3x) = 10\mathcal{L}(\cos(t))$$

$$\mathcal{L}(x'') + 4\mathcal{L}(x') + 3\mathcal{L}(x) = 10\mathcal{L}(\cos(t))$$

$$[s^2X(s) - sx(0) - x'(0)] + 4[sX(s) - x(0)] + 3X(s) = \frac{10s}{s^2+1}$$

$$(s^2 + 4s + 3)X(s) - 2s - 11 = \frac{10s}{s^2+1}$$

$$X(s)(s+1)(s+3) = \frac{10s}{s^2+1} + \frac{(2s+11)(s^2+1)}{(s^2+1)}$$

$$X(s) = \frac{2s^3 + 11s^2 + 12s + 11}{(s^2+1)(s+1)(s+3)}$$

Now use partial fractions

$$\frac{2s^3 + 11s^2 + 12s + 11}{(s^2+1)(s+1)(s+3)} = \frac{A}{s+1} + \frac{B}{s+3} + \frac{Cs+D}{s^2+1}$$

Solving for A, B, C, and D, yields $A = 2$, $B = -1$, $C = 1$, and $D = 2$.

Thus $X(s) = \dfrac{2}{s+1} - \dfrac{1}{s+3} + \dfrac{s+2}{s^2+1}$. Therefore, $x(t) = 2e^{-t} - e^{-3t} + \cos(t) + 2\sin(t)$.

7. Solve the initial value problem

$$x'' + 2x' + 3x = 9t, \quad x(0) = 1, \quad x'(0) = 2.$$

$$\mathcal{L}(x'' + 2x' + 3x) = 9\mathcal{L}(t)$$

$$\mathcal{L}(x'') + 2\mathcal{L}(x') + 3\mathcal{L}(x) = 9\mathcal{L}(t)$$

$$[s^2X(s) - sx(0) - x'(0)] + 2[sX(s) - x(0)] + 3X(s) = \frac{9}{s^2}$$

$$(s^2 + 2s + 3)X(s) - s - 4 = \frac{9}{s^2}$$

$$X(s) = \frac{s^3 + 4s^2 + 9}{(s^2)(s^2 + 2s + 3)}$$

Now use partial fractions

$$\frac{s^3 + 4s^2 + 9}{(s^2)(s^2 + 2s + 3)} = \frac{A}{s} + \frac{B}{s^2} + \frac{Cs+D}{s^2+2s+3}$$

Solving for A, B, C, and D, yields $A = -2$, $B = 3$, $C = 3$, and $D = 5$.

Thus $X(s) = \dfrac{-2}{s} - \dfrac{3}{s^2} + \dfrac{3s+5}{s^2+2s+3}$.

Completing the square of $s^2+2s+3 = (s+1)^2 + \sqrt{2}^2$. So $\dfrac{3s+5}{s^2+2s+3} = \dfrac{3s+5}{(s+1)^2+\sqrt{2}^2} = 3\dfrac{s+1}{(s+1)^2+\sqrt{2}^2} + \sqrt{2}\dfrac{\sqrt{2}}{(s+1)^2+\sqrt{2}^2}$.

Thus, $X(s) = \dfrac{-2}{s} - \dfrac{3}{s^2} + 3\dfrac{s+1}{(s+1)^2+\sqrt{2}^2} + \sqrt{2}\dfrac{\sqrt{2}}{(s+1)^2+\sqrt{2}^2}$. Therefore, $x(t) = -2 + 3t + 3e^{-t}\cos(\sqrt{2}t) + \sqrt{2}e^{-t}\sin(\sqrt{2}t)$.

8. Let $f(t) = \begin{cases} t & \text{if } 0 \le t \le 2 \\ 0 & \text{if } t > 2 \end{cases}$. Solve the initial value problem

$$x'' + x = f(t), \quad x(0) = 0, \quad x'(0) = 0.$$

Hint: First rewrite $f(t)$ using the Heaviside function so that you can take the Laplace transform of both sides.

First note that $f(t) = t + u(t-2)[-t]$. Then we know

$$\mathcal{L}(x'') + \mathcal{L}(x) = \mathcal{L}(t) - \mathcal{L}(u(t-2)\cdot t)$$

$$s^2X(s) + X(s) = \frac{1}{s^2} - e^{-2s}G(s)$$

where $G(s) = \mathcal{L}(t+2) = \dfrac{1}{s^2} + \dfrac{2}{s}$.

Thus,

$$X(s) = \frac{1}{s^2(s^2+1)} - e^{-2s}\frac{1}{s^2(s^2+1)} - 2e^{-2s}\frac{1}{s(s^2+1)}$$

Using partial fractions we find that $\dfrac{1}{s^2(s^2+1)} = \dfrac{1}{s^2} - \dfrac{1}{s^2+1}$ and $\dfrac{1}{s(s^2+1)} = \dfrac{1}{s} - \dfrac{s}{s^2+1}$. This means that

$$X(s) = \frac{1}{s^2} - \frac{1}{s^2+1} - e^{-2s}\frac{1}{s^2} + e^{-2s}\frac{1}{s^2+1} - 2e^{-2s}\frac{1}{s} + 2e^{-2s}\frac{s}{s^2+1}$$

Therefore,

$x(t) = t - \sin(t) - u(t-2)[t-2] + u(t-2)\sin(t-2) - 2u(t-2) + 2u(t-2)\cos(t-2) = t - \sin(t) + u(t-2)[\sin(t-2) + 2\,c$

11.8 JNB Lab: Differential Equations

11.8.1 Chapter Review

Note

This first section of practice problems can serve as a review of the topics covered in the Differential Equations chapter organized by section; most of these questions can (and should) be completed by hand. It is followed by a series of problems drawing from the material of the chapter overall, which should be completed using code.

Section 2 Questions

1. List the type, order, and parameters of each of the following differential equations. For each ordinary differential equation, determine if it is linear or nonlinear.
   - $\frac{dy}{dx} + 3x^2 = sin(x) + 3$
   - $(\frac{dy}{dx})^2 + 3x^2 = sin(x) + 3$
   - $\frac{dy}{dx}k + 3x^2k = y\,sin(x) + 3$
   - $\frac{d^2y}{dx^2} + 3x^2 = sin(x) + 3$

2. List the type, order, and parameters of each of the following differential equations. For each ordinary differential equation, determine if it is linear or nonlinear.
   - $\frac{\partial w}{\partial u} + \frac{\partial^2 w}{\partial v^2} = u$
   - $\frac{\partial w}{\partial u} + \frac{\partial w}{\partial v} = u^2$
   - $\frac{dt}{dv}t = tan(t) + 9$
   - $\frac{dt}{dv}v = tan(t) + 9$

3. Show that $y = e^{2t}$ is a solution for all t to $y'' + y = 5y$.
4. Show that $y = te^{5t}$ is a solution for all t to $y'' - 10y' = -25y$.
5. Show that $y = sin(3t) + cos(3t) + t^3 - 6$ is a solution for all t to $y''' + 9y' = 27t^2 + 6$.
6. Show that $y = 12t^2 + t^4$ is a solution for all t to $(y' + y'') - (y''' + y^{(4)}) = y - t^4 + 4t^3$.

Section 3 Questions

1. Solve the separable equations:
   - $\frac{dx}{dt} = \frac{1}{t}e^{-2x}$
   - $\frac{dx}{dt} = \frac{-1}{x^2t^2}$
   - $\frac{dy}{dx} = cos^2(y)e^{x^2}2x$
   - $\frac{1}{x} - \frac{dx}{dt} = x^{-1}\frac{1}{sec(t)}$
2. Solve the linear equations using the method of integrating factors ($\mu(t) = e^{P(t)}$):
   - $t\frac{dx}{dt} = sin(t) - x$
   - $\frac{dx}{dt} + x4t^3 = \frac{1}{te^{t^4}}$

- $\frac{dy}{dx} + \frac{y}{x} = \frac{1}{x + x^3}$
- $\frac{dy}{dx}x + \frac{-2y}{x^2} = xe^{-x^{-2}}sec(x)tan(x)$

3. Find the equilibrium solutions and draw the phase lines for the following functions. Classify each equilibrium solution.
   - $y' = (y - 6)(y + 7)$
   - $y' = y^3 - 8y^2 + 12y$
   - $y' = y(y^2 - 1)$
   - $y' = y^2(y^2 - 1)$

Section 4 Questions

1. Solve the initial value problem:
   - $x'' - x' - 42x = 0; x'(0) = 0, x''(0) = 1$
   - $x'' - 14x' + 48x = 0; x'(0) = 0, x''(0) = 5$
   - $x'' + 6x' + 9x = 0, x'(0) = 0, x''(0) = 9$
   - $x'' + 16x' + 64x = 0, x'(0) = 0, x''(0) = 8$
   - $x'' + 4x' + 8x = 0, x'(0) = 0, x''(0) = 4$
   - $x'' + 10x' + 30x = 0, x'(0) = 0, x''(0) = 30$
2. Solve the non-homogeneous equations:
   - $x'' - 9x' + 20x = -10$
   - $x'' - 9x' + 20x = 20t^2$
   - $x'' - 9x' + 20x = e^{3t}$
   - $x'' - 9x' + 20x = e^{5t}$
3. Solve the non-homogeneous equations:
   - $x'' + 4x' + 4x = sin(t)$
   - $x'' + 4x' + 4x = 8t^2$

Section 5 Questions

1. Rewrite as a system of first order equations:
   - $x''' + 3x'' + 12x' + x = sin(t) + e^{3t}$
2. Test the following sets of vectors for linear independence:
   - $\left\{\begin{bmatrix}2e^t \\ e^{12t}\end{bmatrix}, \begin{bmatrix}12e^{2t} \\ 6e^{12t}\end{bmatrix}\right\}$
   - $\left\{\begin{bmatrix}5e^{3t} \\ 6e^{6t}\end{bmatrix}, \begin{bmatrix}10e^{3t} \\ 12e^{6t}\end{bmatrix}\right\}$
   - $\left\{\begin{bmatrix}3t \\ 1\end{bmatrix}, \begin{bmatrix}3t \\ tan^2(t)\end{bmatrix}, \begin{bmatrix}6t \\ sec^2(t)\end{bmatrix}\right\}$

3. Find the eigenvalues and eigenvectors:
   - $\begin{bmatrix} 3 & 0 \\ 0 & 4 \end{bmatrix}$
   - $\begin{bmatrix} 6 & 2 \\ -2 & 2 \end{bmatrix}$
   - $\begin{bmatrix} 3 & 2 \\ -9 & -3 \end{bmatrix}$
4. Solve the linear systems where $x' = Ax$ of differential equations, for each choice of A:
   - $A = \begin{bmatrix} 3 & -2 \\ 4 & 9 \end{bmatrix}$
   - $A = \begin{bmatrix} 5 & -2 \\ 2 & 1 \end{bmatrix}$
   - $A = \begin{bmatrix} 6 & 13 \\ -4 & -6 \end{bmatrix}$
5. Find the nullclines, equilibrium solutions, and linearization of the system:
   - $x' = x(1-x) - 4xy; y' = y(2-y) - 3xy$

Section 6 Questions

1. Solve the following Laplace Transforms by hand, using the formula $\mathcal{L}(f(t)) = \int_0^\infty e^{-st} f(t) \mathrm{d}t \equiv F(s)$.
   - $f(t) = sin(t)$
   - $f(t) = cos(t)$
   - $f(t) = 8t + e^{3t} + cos(3t)$
2. Find the inverse Laplace Transforms of the following functions:
   - $\mathcal{L}(x) = -\frac{14}{s^3} + \frac{1}{s^2 + 49}$
   - $\mathcal{L}(x) = \frac{s+9}{(s+2)(s+3)}$
   - $\mathcal{L}(x) = x'' + 12x = 3, x(0) = x'(0) = 0$
3. Express the following functions in terms of the unit step function, then take the Laplace Transform:
   - $f(t) = \begin{cases} 0 & \text{if } t < 2 \\ 1 & \text{if } t > 2 \end{cases}$
   - $f(t) = \begin{cases} 2 & \text{if } 0 < t < 1 \\ 3 & \text{if } 1 < t < 2 \\ 4 & \text{if } t > 2 \end{cases}$

11.8.2 Python Questions

```
import sympy as sp
```

Note

Now that we have practiced solving differential equations by hand, we review some tools and implement new ones to solve differential equations using Python code.

First and second order equations

First and second order equations can be solved fairly easily by using Sympy:

Example: Solve $\frac{dy}{dx} = 3x^2\, y$

```
x = symbols("x") #Independent variable
y = Function("y") #Dependent variable
```

```
ex = 3*x**2 *y(x) - y(x).diff(x) #Enter the function
ex
```

$$3x^2 y(x) - \frac{d}{dx} y(x)$$

```
dsolve(ex) #Solve the equation
```

$$y(x) = C_1 e^{x^3}$$

1. Solve $\frac{dy}{dx} = 3x^2 y$ using Sympy code
2. Solve $y'' + 3y = 0$
3. Solve $\frac{dy}{dx} = cos(x) + 3y$
4. Solve $x'' + 2x' + 2x = 12$
5. Solve $\frac{dy}{dx} = \frac{1}{(1+x^2)} y^2$

Direction Fields

Use Python code to plot the direction fields:

1. $\frac{dy}{dx} = x^2 - y^2 - y$
2. $\frac{dy}{dx} = x(y - x)$
3. $\frac{dy}{dx} = y(x - y)$

Phase Planes and Phase Portraits

Edit the code from Section 5.4 to plot the phase planes, then find the equilibrium solutions:

1. $\frac{dx}{dt} = x(x+3) - 2xy$, $\frac{dy}{dt} = y(3y-8) - xy$
2. $\frac{dx}{dt} = x(5-2x) + xy$, $\frac{dy}{dt} = y(y+1) + y$ Edit the code from Section 5.4 to plot the phase portraits of the same systems
3. $\frac{dx}{dt} = x(x+3) - 2xy$, $\frac{dy}{dt} = y(3y-8) - xy$
4. $\frac{dx}{dt} = x(5-2x) + xy$, $\frac{dy}{dt} = y(y+1) + y$

Nullclines

Plot the nullclines for the following system:

- $x' = x(1-x) - 4xy$; $y' = y(2-y) - 3xy$

Laplace Transforms

Note

Laplace Transforms can also be solved using code- we will utilize both the Sympy and Numpy libraries

Example: Find the Laplace Transform of $f(t) = sin(t)$

```
import numpy as np
import matplotlib.pyplot as plt
import sympy as sym
from sympy.abc import s,t
```

```
f = sym.sin(t)  # define your function with respect to t
F = sym.laplace_transform(f, t, s)  # sym.laplace_transform_(function, func_var, out_
↪var)

print(F)  # returns F(s) and convergence interval
```

```
(1/(s**2 + 1), 0, True)
```

Example: Find the Inverse Laplace Transform of $f(t) = \frac{1}{s^2+1}$

```
# Define the inverse Laplace transform function
def invL(F):
    return sym.inverse_laplace_transform(F, s, t)

F= 1/(s**2+1)
print(invL(F))
```

```
sin(t)*Heaviside(t)
```

Note: Heaviside(t) means that $t \geq 0$

Example: Solve the initial value problem

$$y'' + 4y' + 3y = 10\cos(t), \quad y(0) = 2, \quad y'(0) = 3.$$

```
from sympy import *
s=sym.symbols('s')
f=sym.Function('s')
F=sym.laplace_transform(f(t),t,s)
g=sym.cos(t)
G=sym.laplace_transform(g,t,s)
print('Transform of cos(t) is', G[0])
# Define the inverse Laplace transform function
def invL(F):
    return sym.inverse_laplace_transform(F, s, t)
# Transform the equation  and solve for Y=L(y)
eq=Eq((s**2) * F - 2*s - 3 + 4 *s* F -4*( 2) + 3* F -10*G[0],0 )
Y = solve(eq, F)
print('Transform of y is',Y)

# Take the inverse transform
print('y=',invL(Y[0]))
```

```
Transform of cos(t) is s/(s**2 + 1)
Transform of y is [(2*s**3 + 11*s**2 + 12*s + 11)/(s**4 + 4*s**3 + 4*s**2 + 4*s +␣
↪3)]
y= ((2*sin(t) + cos(t))*exp(3*t) + 2*exp(2*t) - 1)*exp(-3*t)*Heaviside(t)
```

1. Find the Laplace Transforms of the following functions:
   - $f(t) = 3t^2 + cos(3t) + 12$
   - $f(t) = \frac{1}{2}sin(t) + 5t$
2. Find the Inverse Laplace Transforms of the following functions:
   - $f(t) = \frac{7}{s} + \frac{720}{s^7} + \frac{1}{(5s^2 + \frac{1}{5})}$
   - $f(t) = \frac{2}{2s-1} + \frac{72}{s^4}$
3. Solve the IVP:
   - $y'' + 2y' + 3y = 9t$, $y(0) = 1$, $y'(0) = 2$

11.9 JNB Lab Solutions

11.9.1 Chapter Review

Section 2 Questions

1. List the type, order, and parameters of each of the following differential equations. For each ordinary differential equation, determine if it is linear or nonlinear.

- $\frac{dy}{dx} + 3x^2 = sin(x) + 3$
  - ODE, first order, linear (remember, the order depends on the highest derivative, not the power of the independent variable)
- $(\frac{dy}{dx})^2 + 3x^2 = sin(x) + 3$
  - ODE, first order, non-linear
- $\frac{dy}{dx}k + 3x^2k = y\,sin(x) + 3$
  - ODE, first order, linear, parameter k
- $\frac{d^2y}{dx^2} + 3x^2 = sin(x) + 3$
  - ODE, second order, linear
- $\frac{\partial w}{\partial u} + \frac{\partial^2 w}{\partial v^2} = u$
  - PDE, second order
- $\frac{\partial w}{\partial u} + \frac{\partial w}{\partial v} = u^2$
  - PDE, first order
- $\frac{dt}{dv}t = tan(t) + 9$
  - ODE, first order, non-linear
- $\frac{dt}{dv}v = tan(t) + 9$
  - ODE, first order, linear

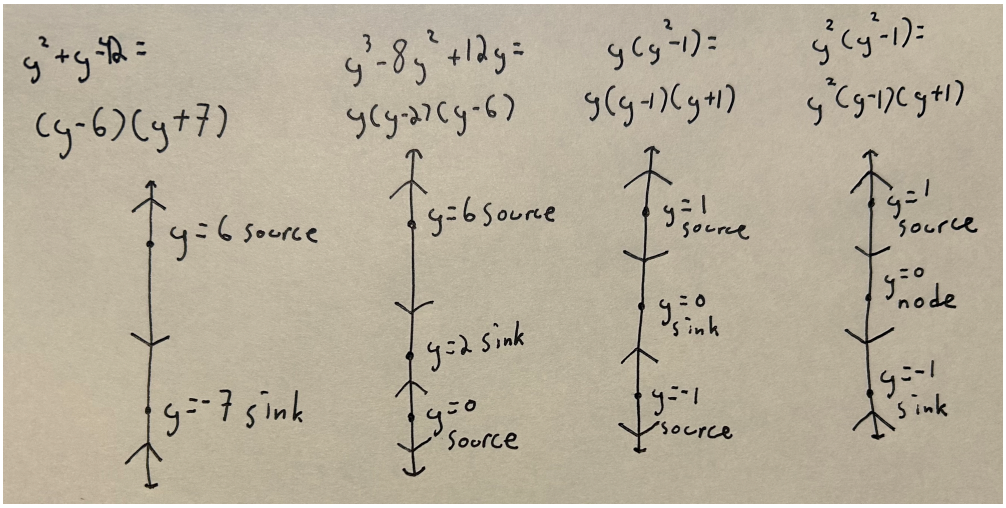
2. Show that $y = e^{2t}$ is a solution for all t to $y'' + y = 5y$.
   - $y' = 2e^{2t}, y'' = 4e^{2t}$; plugging these in, we get $4e^{2t} + e^{2t} = 5e^{2t}$, which is true.
3. Show that $y = te^{5t}$ is a solution for all t to $y'' - 10y' = -25y$.
   - $y' = 5te^{5t} + e^{5t}, y'' = 25te^{5t} + 10e^{5t}$; plugging these in, we get $25te^{5t} + 10e^{5t} - 10(5te^{5t} + e^{5t}) = -25(te^{5t})$. Simplifying the left side will yield $-25te^{5t}$, which is the same as the right side.
4. Show that $y = sin(3t) + cos(3t) + t^3 - 6$ is a solution for all t to $y''' + 9y' = 27t^2 + 6$.
   - $y' = 3cos(3t) - 3sin(3t) + 3t^2, y'' = -9sin(3t) - 9cos(3t) + 6t, y''' = -27cos(3t) + 27sin(3t) + 6$. Plugging these values into the equation yields: $-27cos(3t)+27sin(3t)+6+9(3cos(3t)-3sin(3t)+3t^2) = 27t^2 + 6$; simplifying the left side gives us $27t^2 + 6$, which is the same as the right side.
5. Show that $y = 12t^2 + t^4$ is a solution for all t to $(y' + y'') - (y''' + y^{(4)}) = y - t^4 + 4t^3$.
   - $y' = 24t + 4t^3, y'' = 24 + 12t^2, y''' = 24t, y^{(4)} = 24$. Plugging these in yields: $24t + 4t^3 + 24 + 12t^2 - (24t + 24) = (12t^2 + t^4) - t^4 + 4t^3$. Both sides reduce to $4t^3 + 12t^2$, so $y = 12t^2 + t^4$ is a solution for all t to the initial equation.

Section 3 Questions

1. Solve the separable equations:
   - $\frac{dx}{dt} = \frac{1}{t}e^{-2x}$
     - $x = \frac{ln(2ln(t) + C)}{2}$
   - $\frac{dx}{dt} = \frac{-1}{x^2t^2}$
     - $x = \sqrt[3]{\frac{3}{t} + C}$
   - $\frac{dy}{dx} = cos^2(y)e^{x^2}2x$
     - $y = arctan(e^{x^2} + C)$
   - $\frac{1}{x} - \frac{dx}{dt} = x^{-1}\frac{1}{sec(t)}$
     - $x = \pm\sqrt{2t - 2sin(t) + C}$
2. Solve the linear equations using the method of integrating factors ($\mu(t) = e^{P(t)}$):
   - $t\frac{dx}{dt} = sin(t) - x$
     - $\mu = e^{ln(t)} = t$
     - $x = \frac{-cos(t) + C}{t}$
   - $\frac{dx}{dt} + x4t^3 = \frac{1}{te^{t^4}}$
     - $\mu = e^{t^4}$
     - $x = \frac{ln(t) + C}{e^{t^4}}$
   - $\frac{dy}{dx} + \frac{y}{x} = \frac{1}{x + x^3}$
     - $\mu = e^{ln(x)} = x$
     - $y = \frac{arctan(x) + C}{x}$
   - $\frac{dy}{dx}x + \frac{-2y}{x^2} = xe^{-x^{-2}}sec(x)tan(x)$
     - $\mu = e^{x^{-2}}$
     - $y = \frac{sec(x) + C}{e^{x^{-2}}}$
3. Find the equilibrium solutions and draw the phase lines for the following functions. Classify each equilibrium solution.
   - $y' = (y - 6)(y + 7)$
     - Solutions: $y = 6, -7$
     - Classification: $y = 6$: source. $y = -7$: sink.

- $y' = y^3 - 8y^2 + 12y$
  - Solutions: $y = 0, 2, 6$
  - Classification: $y = 0$: source. $y = 2$: sink. $y = 6$: source.
- $y' = y(y^2 - 1)$
  - Solutions: $y = 0, 1, -1$
  - Classification: $y = -1$: source. $y = 0$: sink. $y = 1$: source.
- $y' = y^2(y^2 - 1)$
  - Solutions: $y = 0, 1, -1$
  - Classification: $y = -1$: sink. $y = 0$: node. $y = 1$: source.

Below are hand-drawn phase lines for all four equations.



Note

Take note of how the repeated root of 0 in the last problem compared with the one before changes the classification of the equilibrium solutions. The phase lines can be drawn by hand or created using the sample code in Section 3.4 of the chapter (Autonomous Equations).

Section 4 Questions

1. Solve the initial value problem:
   - $x'' - x' - 42x = 0; x'(0) = 0, x''(0) = 1$
     - $x = \dfrac{e^{7t}}{91} + \dfrac{e^{-6t}}{78}$
   - $x'' - 14x' + 48x = 0; x'(0) = 0, x''(0) = 5$
     - $x = \dfrac{5}{16}e^{8t} - \dfrac{5}{12}e^{6t}$
   - $x'' + 6x' + 9x = 0, x'(0) = 0, x''(0) = 9$
     - $x = -e^{-3t} - 3e^{-3t}t$

- $x'' + 16x' + 64x = 0, x'(0) = 0, x''(0) = 8$
  - $x = -\frac{e^{-8t}}{8} - e^{-8t}t$
- $x'' + 4x' + 8x = 0, x'(0) = 0, x''(0) = 4$
  - $x = -\frac{e^{-2t}}{2}(cos(2t) + sin(2t))$
- $x'' + 10x' + 30x = 0, x'(0) = 0, x''(0) = 30$
  - $x = e^{-5t}(-cos(\sqrt{5}t) - \sqrt{5}sin(\sqrt{5}t))$

2. Solve the non-homogeneous equations:
   - $x'' - 9x' + 20x = -10$
     - $c_1e^{5t} + c_2e^{4t} - \frac{1}{2}$
   - $x'' - 9x' + 20x = 20t^2$
     - $c_1e^{5t} + c_2e^{4t} + t^2 + \frac{9t}{10} + \frac{61}{200}$
   - $x'' - 9x' + 20x = e^{3t}$
     - $c_1e^{5t} + c_2e^{4t} + \frac{1}{2}e^{3t}$
   - $x'' - 9x' + 20x = e^{5t}$
     - $c_1e^{5t} + c_2e^{4t} + te^{5t}$
3. Solve the non-homogeneous equations:
   - $x'' + 4x' + 4x = sin(t)$
     - $c_1e^{-2t} + c_2e^{-2t}t + \frac{3}{25}sin(t) - \frac{4}{25}cos(t)$
   - $x'' + 4x' + 4x = 8t^2$
     - $c_1e^{-2t} + c_2e^{-2t}t + 2t^2 - 4t + 3$

Section 5 Questions

1. Rewrite as a system of first order equations:
   - $x''' + 3x'' + 12x' + x = sin(t) + e^{3t}$
     - Define new variables: $x_1 = x$, $x_2 = x'$, $x_3 = x''$
     - From this, we see that $x_1' = x_2$, $x_2' = x_3$, and so we can rewrite the original equation as:
     - Solution: $x_3' = -x_1 - 12x_2 - 3x_3 + sin(t) + e^{3t}$
2. Test the following sets of vectors for linear independence:
   - $\left\{\begin{bmatrix}2e^t\\e^{12t}\end{bmatrix}, \begin{bmatrix}12e^{2t}\\6e^{12t}\end{bmatrix}\right\}$
     - independent
   - $\left\{\begin{bmatrix}5e^{3t}\\6e^{6t}\end{bmatrix}, \begin{bmatrix}10e^{3t}\\12e^{6t}\end{bmatrix}\right\}$
     - dependent

- $\left\{\begin{bmatrix}3t\\1\end{bmatrix}, \begin{bmatrix}3t\\tan^2(t)\end{bmatrix}, \begin{bmatrix}6t\\sec^2(t)\end{bmatrix}\right\}$
  - dependent: $1 + tan^2(t) = sec^2(t)$, $so, v_1 + v_2 - v_3 = 0$

3. Find the eigenvalues and eigenvectors:

- $\begin{bmatrix}3 & 0\\0 & 4\end{bmatrix}$
  - Eigenvalues: $r_1 = 3, r_2 = 4$
  - Eigenvectors: $v_1 = \begin{bmatrix}1\\0\end{bmatrix}, v_2 = \begin{bmatrix}0\\1\end{bmatrix}$
- $\begin{bmatrix}6 & 2\\-2 & 2\end{bmatrix}$
  - Eigenvalue: $r = 4$
  - Eigenvector: $v = \begin{bmatrix}-1\\1\end{bmatrix}$
- $\begin{bmatrix}3 & 2\\-9 & -3\end{bmatrix}$
  - Eigenvalues: $r_1 = 3i, r_2 = -3i$
  - Eigenvectors: $v_1 = \begin{bmatrix}\frac{-1-i}{3}\\1\end{bmatrix}, v_2 = \begin{bmatrix}\frac{-1+i}{3}\\1\end{bmatrix}$

4. Solve the linear systems where $x' = Ax$ of differential equations, for each choice of A:

- $A = \begin{bmatrix}3 & -2\\4 & 9\end{bmatrix}$
  - Eigenvalues: $r_1 = 5, r_2 = 7$
  - Eigenvectors: $v_1 = \begin{bmatrix}-1\\1\end{bmatrix}, v_2 = \begin{bmatrix}-\frac{1}{2}\\1\end{bmatrix}$
  - Solution: $x = c_1e^{5t}\begin{bmatrix}-1\\1\end{bmatrix} + c_2e^{7t}\begin{bmatrix}-1\\2\end{bmatrix}$
- $A = \begin{bmatrix}5 & -2\\2 & 1\end{bmatrix}$
  - Eigenvalue: $r = 3$
  - Eigenvector: $\begin{bmatrix}1\\1\end{bmatrix}$
  - Solution: $x = c_1e^{3t}\begin{bmatrix}1\\1\end{bmatrix} + c_2e^{3t}(t\begin{bmatrix}1\\1\end{bmatrix} + \begin{bmatrix}\frac{1}{2}\\0\end{bmatrix})$
- $A = \begin{bmatrix}6 & 13\\-4 & -6\end{bmatrix}$
  - Eigenvalues: $\pm 4i$
  - Eigenvectors: for $+4i$: $v_1 = \begin{bmatrix}\frac{-3-2i}{2}\\1\end{bmatrix}$; for $-4i$: $v_2 = \begin{bmatrix}\frac{-3+2i}{2}\\1\end{bmatrix}$

  - Solution: $x = c_1(cos(4t)\begin{bmatrix}-3\\2\end{bmatrix} - sin(4t)\begin{bmatrix}-2\\0\end{bmatrix}) + c_2(cos(4t)\begin{bmatrix}-2\\0\end{bmatrix} + sin(4t)\begin{bmatrix}-3\\2\end{bmatrix})$

5. Find the nullclines, equilibrium solutions, and linearization of the system:
   - $x' = x(1-x) - 4xy; y' = y(2-y) - 3xy$
     - x' nullclines: $x = 0$, $y = \frac{1-x}{4}$
     - y' nullclines: $y = 0$, $y = 2 - 3x$
     - Equilibrium solutions: $(0,0), (1,0), (0,2), (\frac{7}{11}, \frac{1}{11})$
     - Linearization: $(0,0)$: $\begin{bmatrix}1 & 0\\0 & 2\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix}$; $(1,0)$: $\begin{bmatrix}-1 & -4\\0 & -1\end{bmatrix}\begin{bmatrix}x-1\\y\end{bmatrix}$; $(0,2)$: $\begin{bmatrix}-7 & 0\\-6 & -2\end{bmatrix}\begin{bmatrix}x\\y-2\end{bmatrix}$; $(\frac{7}{11}, \frac{1}{11})$: $\begin{bmatrix}-\frac{7}{11} & -\frac{28}{11}\\-\frac{3}{11} & -\frac{1}{11}\end{bmatrix}\begin{bmatrix}x-\frac{7}{11}\\y-\frac{1}{11}\end{bmatrix}$

Section 6 Questions

1. Solve the following Laplace Transforms by hand, using the formula $\mathcal{L}(f(t)) = \int_0^\infty e^{-st} f(t)\mathrm{d}t \equiv F(s)$.
   - $f(t) = sin(t)$
     - $\frac{1}{s^2+1}$
   - $f(t) = cos(t)$
     - $\frac{s}{s^2+1}$
   - $f(t) = 8t + e^{3t} + cos(3t)$
     - $\frac{8}{s^2} + \frac{1}{s-3} + \frac{s}{s^2+9}$
2. Find the inverse Laplace Transforms of the following functions:
   - $\mathcal{L}(x) = -\frac{14}{s^3} + \frac{1}{s^2+49}$
     - $-7t^2 + \frac{1}{7}sin(7t)$ (Hint: you can multiply $\frac{1}{s^2+49}$ by 7 in the numerator, then by $\frac{1}{7}$ out front to get it into the right form to use the formula for $sin(t)$)
   - $\mathcal{L}(x) = \frac{s+9}{(s+2)(s+3)}$
     - Partial fractions: $\frac{s+9}{(s+2)(s+3)} = \frac{A}{s+2} + \frac{B}{s+3}$; $A = 7, B = -6$. Therefore, $\mathcal{L}(x) = 7e^{-2t} - 6e^{-3t}$
   - $\mathcal{L}(x) = x'' + 12x = 3, x(0) = x'(0) = 0$

Take the Laplace Transform of both sides:

$$s^2X(s) - sx(0) - x'(0) + 12X(s) = \frac{3}{s}$$

Solve for $X(s)$:

$$X(s) = \frac{3}{s(s^2+12)}$$

Partial Fraction Decomposition:

$$\frac{3}{s(s^2+12)} = \frac{A}{s} + \frac{Bs+C}{s^2+12}; A = \frac{1}{4}, B = -\frac{1}{4}, C = 0$$

$$\frac{1}{4s} - \frac{1}{4}(\frac{s}{s^2+12})$$

Inverse Laplace Transform:

$$\frac{1}{4} - \frac{1}{4}cos(\sqrt{12}t)$$

3. Express the following functions in terms of the unit step function, then take the Laplace Transform:

- $f(t) = \begin{cases} 0 & \text{if } t < 2 \\ 1 & \text{if } t > 2 \end{cases}$
  - $u(t-2); \mathcal{L}(x) = \frac{e^{-2s}}{s}$
- $f(t) = \begin{cases} 2 & \text{if } 0 < t < 1 \\ 3 & \text{if } 1 < t < 2 \\ 4 & \text{if } t > 2 \end{cases}$
  - $2 + u(t-1) + u(t-2); \mathcal{L}(x) = \frac{2}{s} + \frac{e^{-s}}{s} + \frac{e^{-2s}}{s}$

11.9.2 Python Questions

```
import sympy as sp
```

First and second order equations

1. Solve $\frac{dy}{dx} = 3x^2y$ using Sympy code

```
x = symbols("x")
y = Function("y")
```

```
eq1 = - y(x).diff(x) + 3* x**2 *y(x)
dsolve(eq1)
```

$$y(x) = C_1e^{x^3}$$

2. Solve $y'' + 3y = 0$

```
t = symbols("t")
y = Function("y")
```

```
eq2 = y(t).diff(t).diff(t)+3*y(t)
dsolve(eq2)
```

$$y(t) = C_1 \sin\left(\sqrt{3}t\right) + C_2 \cos\left(\sqrt{3}t\right)$$

3. Solve $\frac{dy}{dx} = cos(x) + 3y$

```
x = symbols("x")
y = Function("y")
```

```
eq3 = - y(x).diff(x) + sp.cos(x) + 3*y(x)
dsolve(eq3)
```

$$y(x) = C_1 e^{3x} + \frac{\sin(x)}{10} - \frac{3\cos(x)}{10}$$

4. Solve $x'' + 2x' + 2x = 12$

```
t = symbols("t")
x = Function("x")
```

```
eq4 = 12 -2*x(t) -2*x(t).diff(t) -x(t).diff(t).diff(t)
dsolve(eq4)
```

$$x(t) = \left(C_1 \sin(t) + C_2 \cos(t)\right) e^{-t} + 6$$

5. Solve $\frac{dy}{dx} = \frac{1}{(1+x^2)} y^2$

```
x = symbols("x")
y = Function("y")
```

```
eq5 = -y(x).diff(x) + y(x)**(2) / (1 + x**2)
dsolve(eq5)
```

$$y(x) = -\frac{1}{C_1 + \operatorname{atan}(x)}$$

Direction Fields

Use Python code to plot the direction fields:

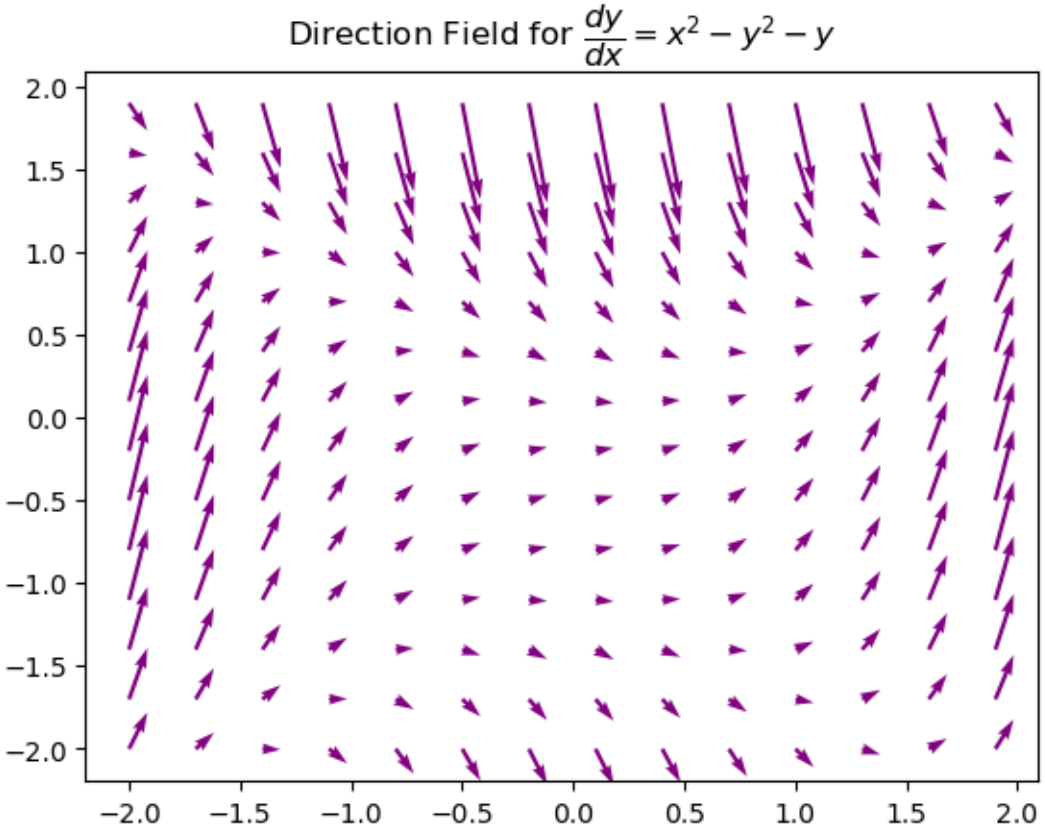
1. $\frac{dy}{dx} = x^2 - y^2 - y$

```
import numpy as np
import matplotlib.pyplot as plt

nx, ny = .3, .3
x = np.arange(-2,2,nx)
y = np.arange(-2,2,ny)
```

(continues on next page)

(continued from previous page)

```
#Create a rectangular grid with points
X,Y = np.meshgrid(x,y)

#Define the functions
dy = X**2 - Y**2 - Y
dx = np.ones(dy.shape)

#Plot
plt.quiver(X,Y,dx,dy,color='purple')
plt.title('Direction Field for $ \dfrac{dy}{dx} = x^2 - y^2 - y $ ')
plt.show()
```



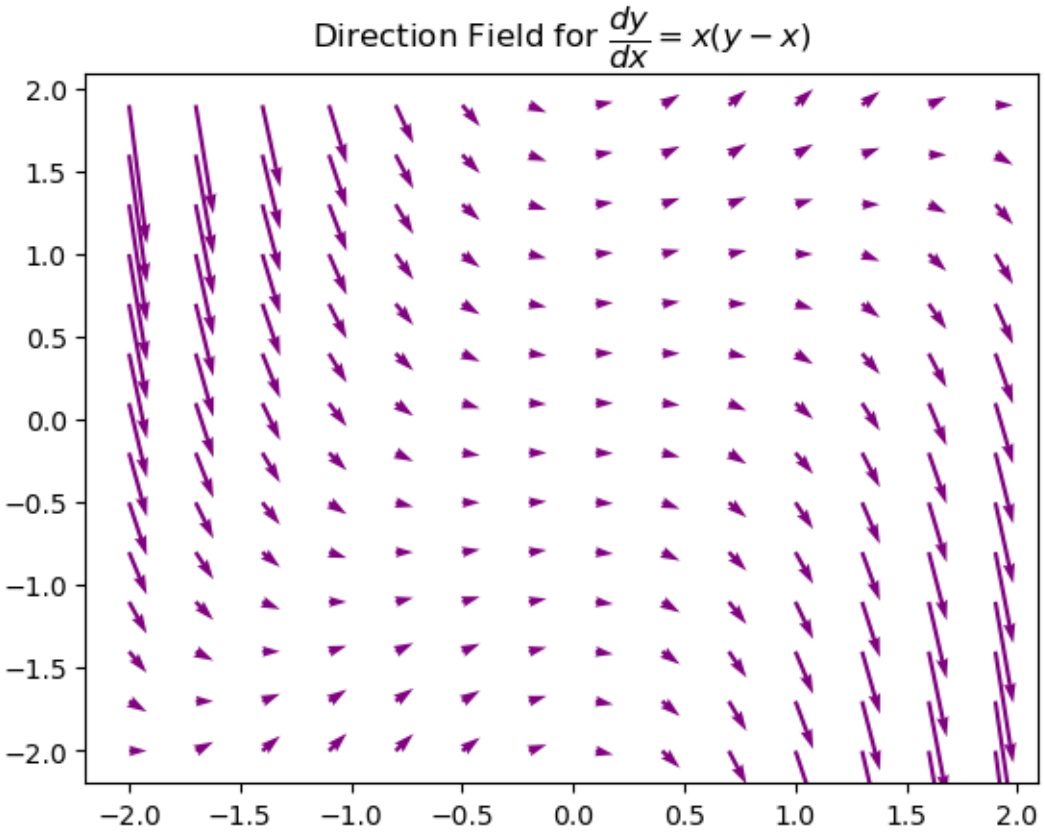
2. $\dfrac{dy}{dx} = x(y - x)$

```
import numpy as np
import matplotlib.pyplot as plt

nx, ny = .3, .3
x = np.arange(-2,2,nx)
y = np.arange(-2,2,ny)

#Create a rectangular grid with points
X,Y = np.meshgrid(x,y)

#Define the functions
```

(continues on next page)

(continued from previous page)

```
dy = X*(Y-X)
dx = np.ones(dy.shape)

#Plot
plt.quiver(X,Y,dx,dy,color='purple')
plt.title('Direction Field for $ \dfrac{dy}{dx} = x(y - x) $')
plt.show()
```



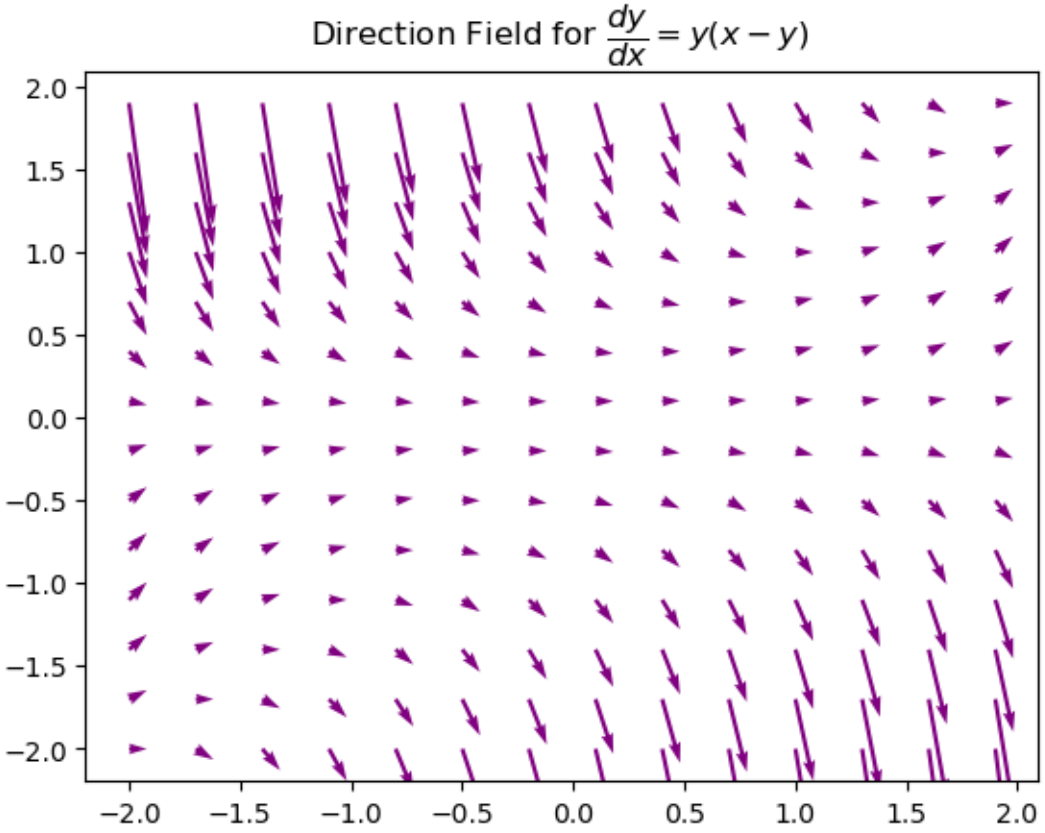
3. $\dfrac{dy}{dx} = y(x-y)$

```
import numpy as np
import matplotlib.pyplot as plt

nx, ny = .3, .3
x = np.arange(-2,2,nx)
y = np.arange(-2,2,ny)

#Create a rectangular grid with points
X,Y = np.meshgrid(x,y)

#Define the functions
dy = Y*(X-Y)
dx = np.ones(dy.shape)

#Plot
plt.quiver(X,Y,dx,dy,color='purple')
```

(continues on next page)

(continued from previous page)

```
plt.title('Direction Field for $ \dfrac{dy}{dx} = y(x - y) $ ')
plt.show()
```



Phase Planes and Phase Portraits

Edit the code from Section 5.4 to plot the phase planes, then find the equilibrium solutions:

1. $\frac{dx}{dt} = x(x+3) - 2xy$, $\frac{dy}{dt} = y(3y-8) - xy$

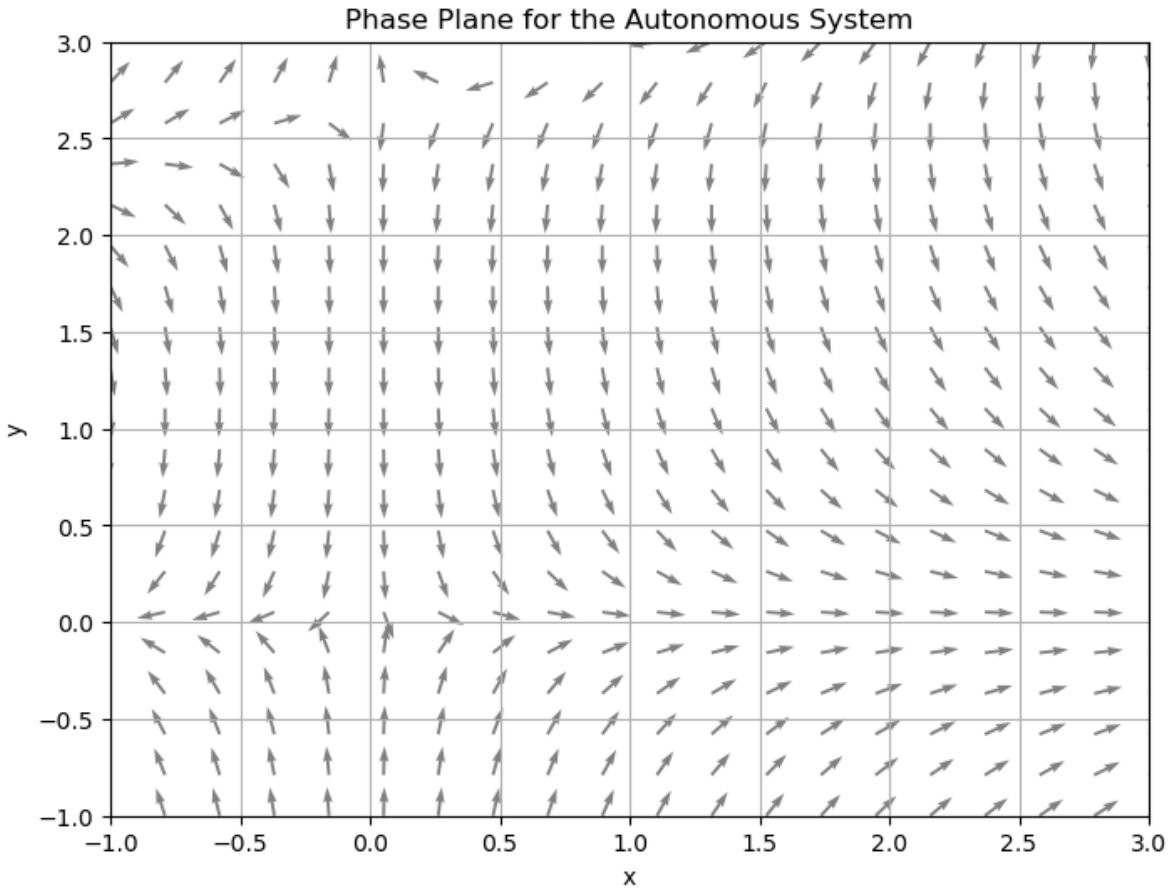
    - Equilibrium solutions: $(0,0), (0,\frac{8}{3}), (-3,0), (7,5)$

```
import numpy as np
import matplotlib.pyplot as plt

# Set the range of values for x and y
x = np.linspace(-1, 3, 20)
y = np.linspace(-1, 3, 20)

# Create a grid of points (x, y)
X, Y = np.meshgrid(x, y)

# Define the system of equations
dx = X*(X + 3) - 2*X*Y
dy = Y*(3*Y - 8) - X*Y
```

(continues on next page)

(continued from previous page)

```
# Normalize the arrows to have the same length
N = np.sqrt(dx**2 + dy**2)
dx = dx/N
dy = dy/N

# Create the quiver plot
plt.figure(figsize=(8, 6))
plt.quiver(X, Y, dx, dy, color='gray')

plt.title('Phase Plane for the Autonomous System')
plt.xlabel('x')
plt.ylabel('y')
plt.xlim([-1, 3])
plt.ylim([-1, 3])
plt.grid(True)
plt.show()
```



2. $\frac{dx}{dt} = x(5-2x) + xy$, $\frac{dy}{dt} = y(y+1) + y$

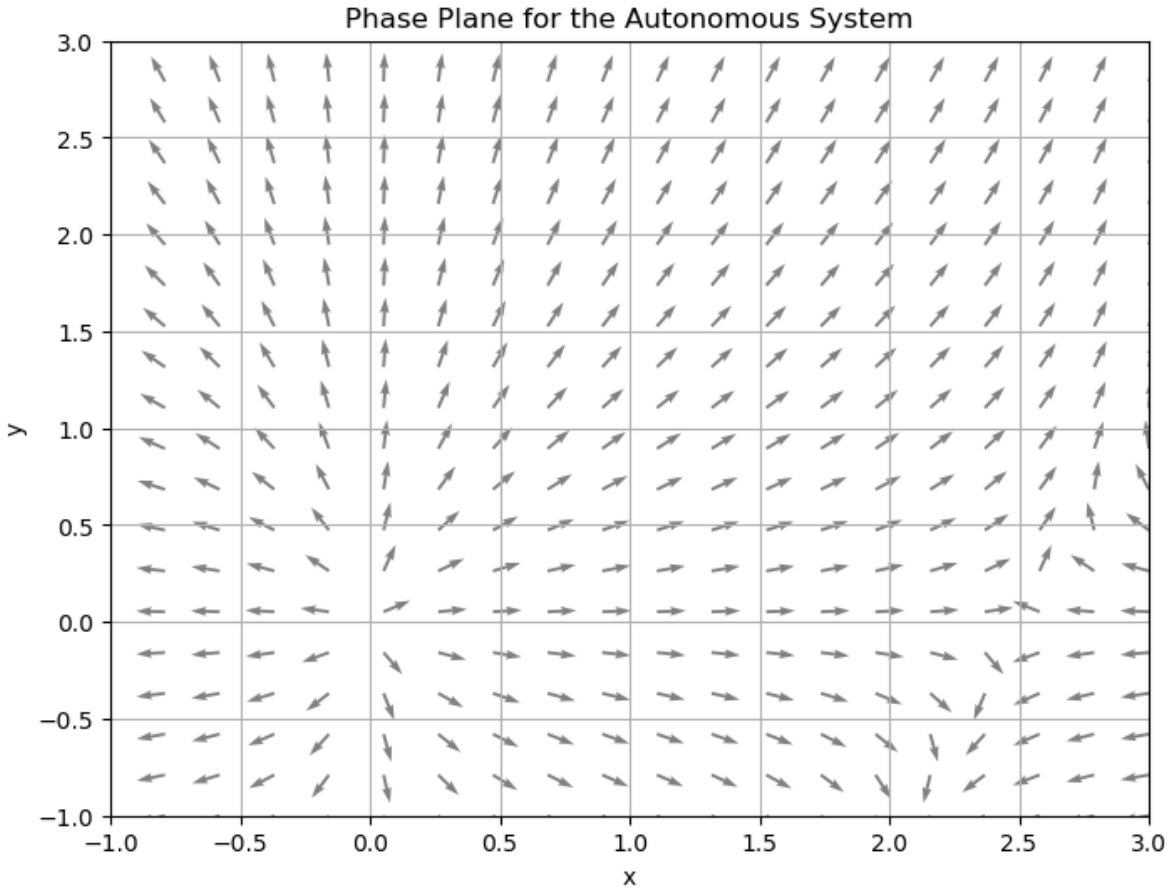
   - Equilibrium solutions: $(0,0), (0,-2), (\frac{5}{2}, 0), (1.5, -2)$

```
import numpy as np
import matplotlib.pyplot as plt
```

(continues on next page)

(continued from previous page)

```
# Set the range of values for x and y
x = np.linspace(-1, 3, 20)
y = np.linspace(-1, 3, 20)

# Create a grid of points (x, y)
X, Y = np.meshgrid(x, y)

# Define the system of equations
dx = X*(5 - 2*X) + X*Y
dy = Y*(Y + 1) + Y

# Normalize the arrows to have the same length
N = np.sqrt(dx**2 + dy**2)
dx = dx/N
dy = dy/N

# Create the quiver plot
plt.figure(figsize=(8, 6))
plt.quiver(X, Y, dx, dy, color='gray')

plt.title('Phase Plane for the Autonomous System')
plt.xlabel('x')
plt.ylabel('y')
plt.xlim([-1, 3])
plt.ylim([-1, 3])
plt.grid(True)
plt.show()
```



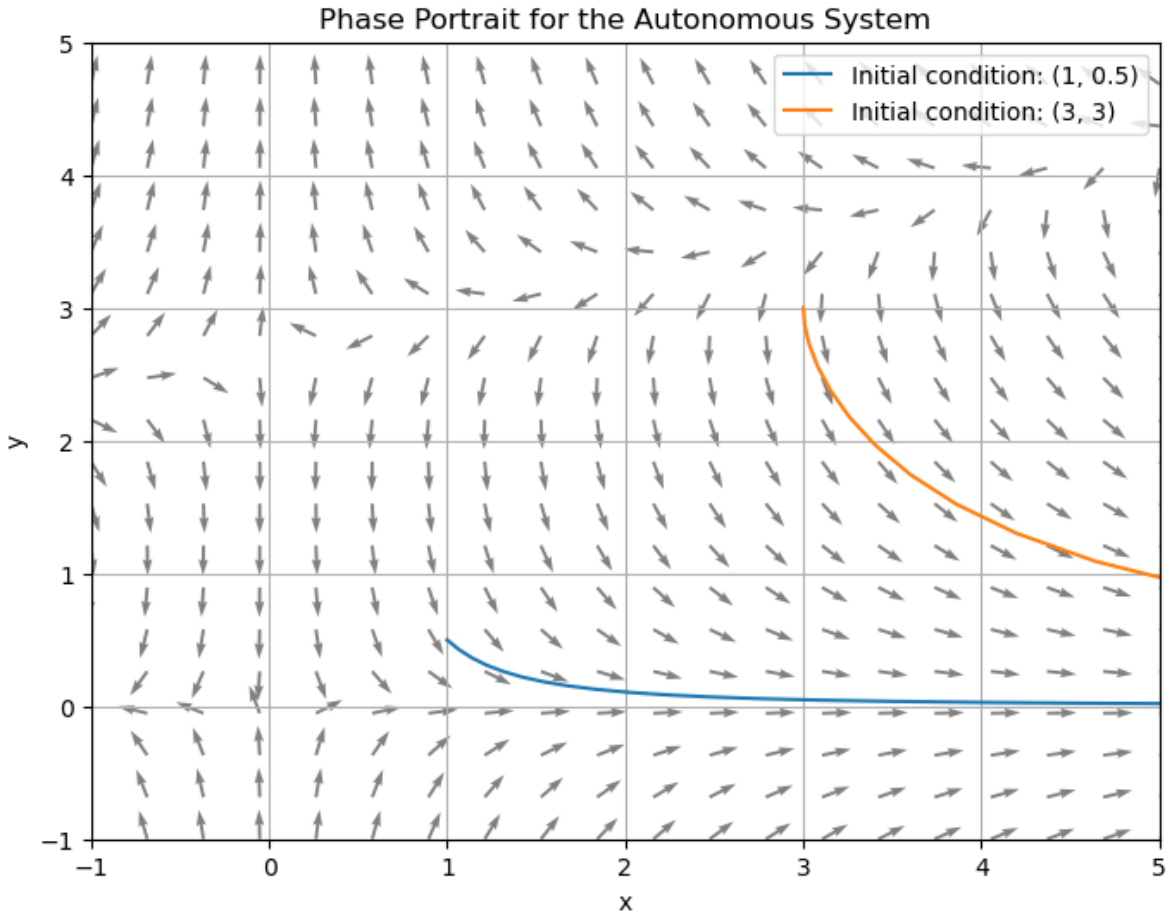
Edit the code from Section 5.4 to plot the phase portraits of the same systems 3. $\frac{dx}{dt} = x(x+3) - 2xy$, $\frac{dy}{dt} = y(3y-8) - xy$

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint

# Define the range of values for x and y
x = np.linspace(-1, 5, 20)
y = np.linspace(-1, 5, 20)

# Create a grid of points (x, y)
X, Y = np.meshgrid(x, y)

# Define the system of equations
U = X*(X + 3) - 2*X*Y
V = Y*(3*Y - 8) - X*Y

# Normalize the arrows to have the same length
N = np.sqrt(U**2 + V**2)
U = U/N
V = V/N

# Define the system for odeint
def dFdt(F, t):
```

(continues on next page)

(continued from previous page)

```
    return [F[0]*(F[0] + 3) - 2*F[0]*F[1], F[1]*(3*F[1] - 8) -F[0]*F[1] ]

# Set the time points where the solution is computed
t = np.linspace(-10, 10, 1000)  # Negative times are included here

# Set a collection of initial conditions
initial_conditions = [(1, 0.5), (3, 3)]

# Create the quiver plot
plt.figure(figsize=(8, 6))
plt.quiver(X, Y, U, V, color='gray')

# Loop over each initial condition and plot the solution curve
for ic in initial_conditions:
    F = odeint(dFdt, ic, t)
    plt.plot(F[:, 0], F[:, 1], label=f'Initial condition: {ic}')

plt.title('Phase Portrait for the Autonomous System')
plt.xlabel('x')
plt.ylabel('y')
plt.xlim([-1, 5])
plt.ylim([-1, 5])
plt.grid(True)
plt.legend()
plt.show()
```



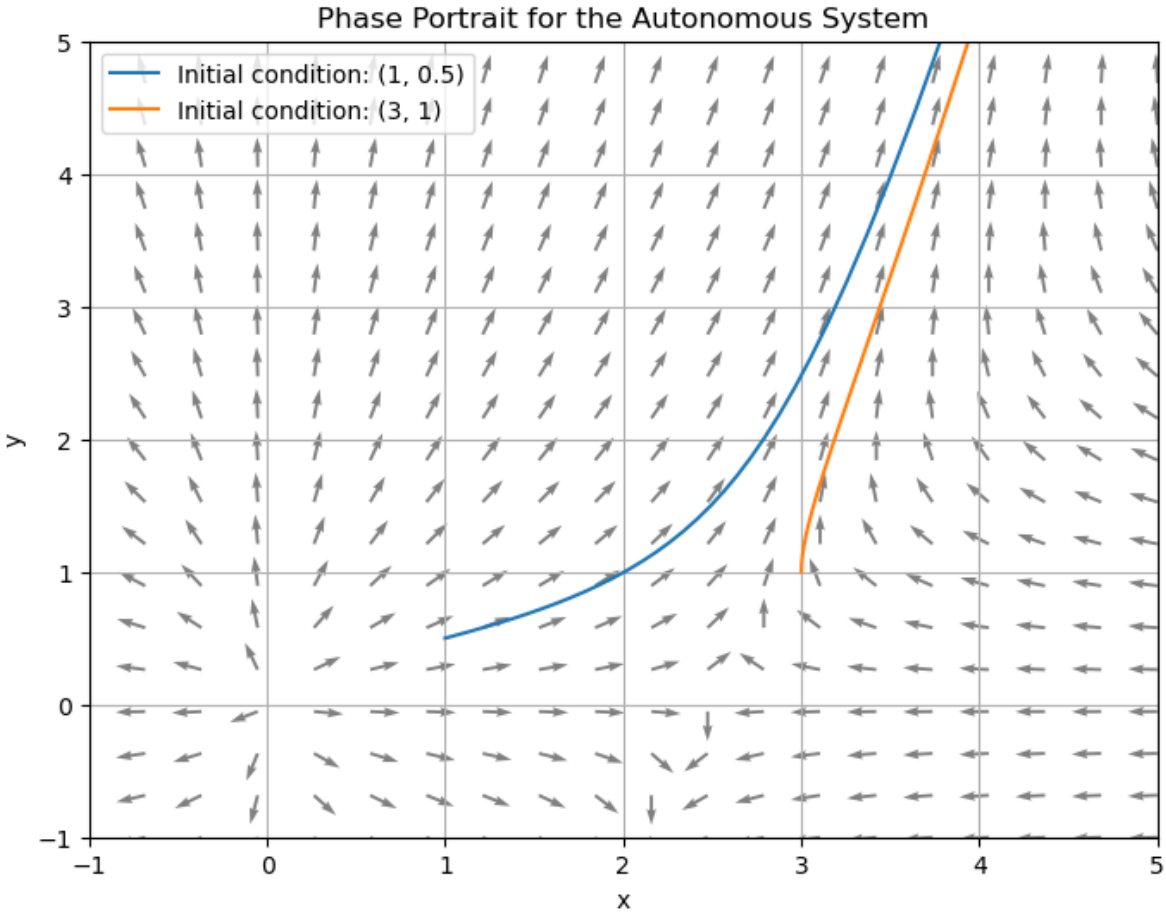
4. $\frac{dx}{dt} = x(5-2x) + xy$, $\frac{dy}{dt} = y(y+1) + y$

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint

# Define the range of values for x and y
x = np.linspace(-1, 5, 20)
y = np.linspace(-1, 5, 20)

# Create a grid of points (x, y)
X, Y = np.meshgrid(x, y)

# Define the system of equations
U = X*(5 - 2*X) + X*Y
V = Y*(Y + 1) + Y

# Normalize the arrows to have the same length
N = np.sqrt(U**2 + V**2)
U = U/N
V = V/N

# Define the system for odeint
def dFdt(F, t):
    return [F[0]*(5 - 2*F[0]) + F[0]*F[1], F[1]*(F[1] + 1) + F[1] ]
```

(continues on next page)

(continued from previous page)

```
# Set the time points where the solution is computed
t = np.linspace(-10, 10, 1000)  # Negative times are included here

# Set a collection of initial conditions
initial_conditions = [(1, 0.5), (3, 1)]

# Create the quiver plot
plt.figure(figsize=(8, 6))
plt.quiver(X, Y, U, V, color='gray')

# Loop over each initial condition and plot the solution curve
for ic in initial_conditions:
    F = odeint(dFdt, ic, t)
    plt.plot(F[:, 0], F[:, 1], label=f'Initial condition: {ic}')

plt.title('Phase Portrait for the Autonomous System')
plt.xlabel('x')
plt.ylabel('y')
plt.xlim([-1, 5])
plt.ylim([-1, 5])
plt.grid(True)
plt.legend()
plt.show()
```



Nullclines

Plot the nullclines for the following system (from Chapter review section 5):

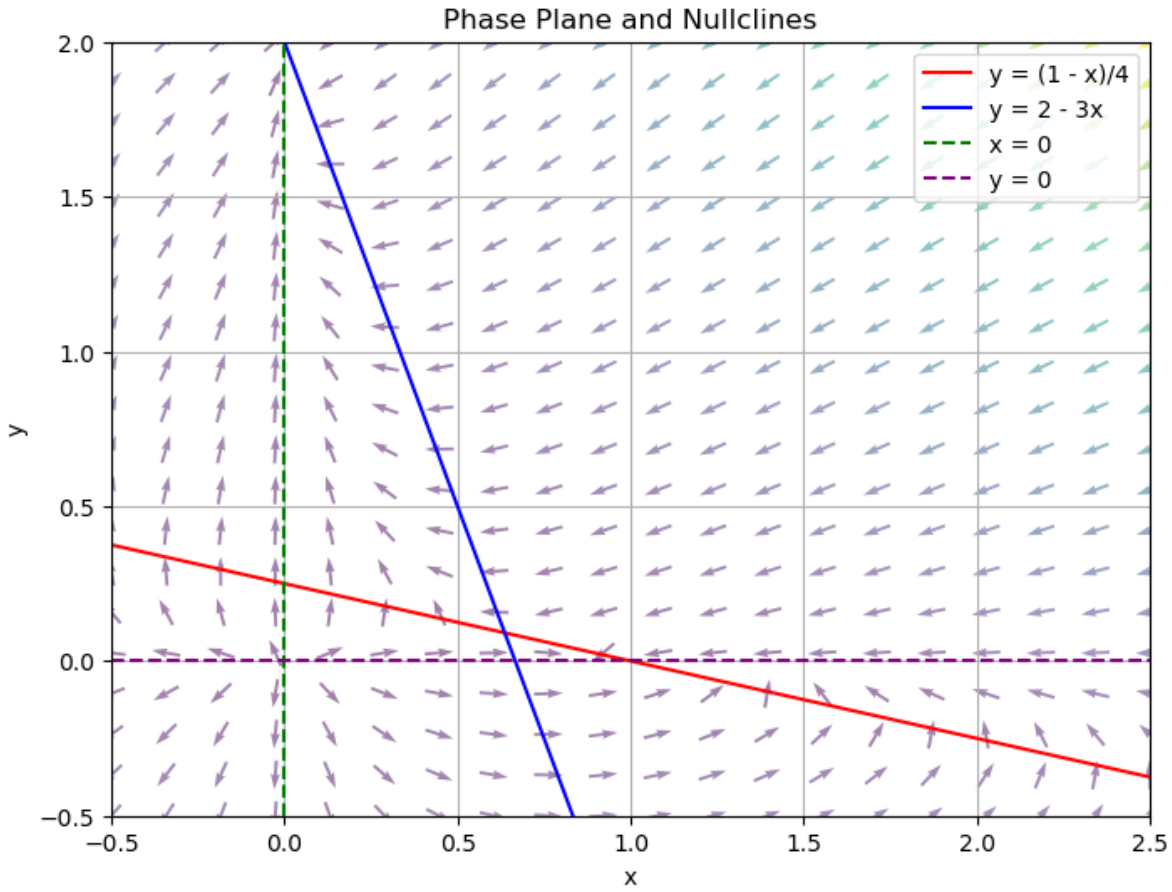
- $x' = x(1-x) - 4xy; y' = y(2-y) - 3xy$

```
import numpy as np
import matplotlib.pyplot as plt

# Define the system of equations
def system(x, y):
    dx = x * (1 - x) - 4 * x * y
    dy = y * (2 - y) - 3 * x * y
    return dx, dy

# Create an x, y grid
y, x = np.mgrid[-0.5:2:20j, -0.5:2.5:20j]

# Calculate dx, dy for the vector field
dx, dy = system(x, y)

# Normalize the arrows so their size represents their speed
M = (np.hypot(dx, dy))
M[M == 0] = 1.
dx /= M
dy /= M

plt.figure(figsize=(8, 6))

# Plot the vector field using quiver
plt.quiver(x, y, dx, dy, M, pivot='mid', alpha=0.5)

# Create an x vector
x = np.linspace(-0.5, 2.5, 200)

# Define nullclines
y1 = (1 - x)/4
y2 = 2 - 3 * x

# Plot nullclines
plt.plot(x, y1, 'r', label="y = (1 - x)/4")
plt.plot(x, y2, 'b', label="y = 2 - 3x")

# Also plot x=0 and y=0
plt.axvline(0, color='green', linestyle='--', label="x = 0")
plt.axhline(0, color='purple', linestyle='--', label="y = 0")

# Labels and legends
plt.xlabel('x')
plt.ylabel('y')
plt.legend()

# Set the x and y axis cutoffs
plt.ylim([-0.5,2])
plt.xlim([-0.5,2.5])

plt.grid(True)
plt.title('Phase Plane and Nullclines')
```

(continues on next page)

(continued from previous page)

```
plt.show()
```



Laplace Transforms

```
import numpy as np
import matplotlib.pyplot as plt
import sympy as sym
from sympy.abc import s,t
```

1. Find the Laplace Transforms of the following functions:

- $f(t) = 3t^2 + cos(3t) + 12$

```
f = 3*t**2 + sym.cos(3*t) + 12  # define your function with respect to t
F = sym.laplace_transform(f, t, s)  # sym.laplace_transform_(function, func_var, out_
↪var)

print(F)  # returns F(s) and convergence interval
```

```
(s/(s**2 + 9) + 12/s + 6/s**3, 0, True)
```

- $f(t) = \frac{1}{2}sin(t) + 5t$

```
f = 1/2 * sym.sin(t) + 5*t  # define your function with respect to t
F = sym.laplace_transform(f, t, s)  # sym.laplace_transform_(function, func_var, out_
↪var)

print(F)  # returns F(s) and convergence interval
```

```
(0.5/(s**2 + 1) + 5/s**2, 0, True)
```

2. Find the Inverse Laplace Transforms of the following functions:

- $f(t) = \frac{7}{s} + \frac{720}{s^7} + \frac{1}{(5s^2 + \frac{1}{5})}$

```
# Define the inverse Laplace transform function
def invL(F):
    return sym.inverse_laplace_transform(F, s, t)

F= 7/s + 720/s**7 + 1/(5*s**2+1/5)
print(invL(F))
```

```
t**6*Heaviside(t) + 1.0*sin(0.2*t)*Heaviside(t) + 7*Heaviside(t)
```

- $f(t) = \frac{2}{2s-1} + \frac{72}{s^4}$

```
# Define the inverse Laplace transform function
def invL(F):
    return sym.inverse_laplace_transform(F, s, t)

F= 2/(2*s-1) + 72/s**4
print(invL(F))
```

```
12*t**3*Heaviside(t) + exp(t/2)*Heaviside(t)
```

3. Solve the IVP:

- $y'' + 2y' + 3y = 9t$, $y(0) = 1$, $y'(0) = 2$

```
import numpy as np
from sympy import *
import sympy as sym
from sympy.abc import s,t

# Define the inverse Laplace transform function
def invL(F):
    return sym.inverse_laplace_transform(F, s, t)
#Solve the IVP
s=sym.symbols('s')
t=sym.symbols('t')
f=sym.Function('s')
F=sym.laplace_transform(f(t),t,s)
g=t
G=sym.laplace_transform(g,t,s)
print('Transform of t is', G[0])

# Transform the equation  and solve for Y=L(y)
eq=Eq((s**2) * F - s - 2 + 2 *s* F -2 + 3* F -9*G[0],0 )
```

(continues on next page)

(continued from previous page)

```
Y = solve(eq, F)
print('Transform of y is',Y)

# Take the inverse transform
print('y=',invL(Y[0]))
```

```
Transform of t is s**(-2)
Transform of y is [(s**3 + 4*s**2 + 9)/(s**2*(s**2 + 2*s + 3))]
y= ((3*t - 2)*exp(t) + sqrt(2)*sin(sqrt(2)*t) + 3*cos(sqrt(2)*t))*exp(-
↪t)*Heaviside(t)
```

CHAPTER 12

DIFFERENTIAL EQUATIONS FOR THE BENEFIT OF SOCIETY - Wheaton College Team

This chapter includes several collaborative projects supervised by Dr. Paul Isihara. Wheaton College student team members are: (CWS Model) Anne Bowen, Isaac Brown, Tim Dennison, Wesley Hanna, and Kyle Williams; (Gravity-fed Water Delivery) Andrew Dornbush, Timothy Dennison, Kristianna Russo, and David Schultz; (Earthquake Resistant Construction) Andrew Gardener; (HIV-AIDS JNB Lab) Andrew Barker, Jordan Hostetter, Michael Hubbell, Kyle Hunsberger, Kimberly Waite, and Rebecca Wilson.

12.1 Introduction

The purpose of this chapter is to show how differential equations can be used to solve real world issues in both human disease tracking and natural phenomena.

- Logistic Growth and COVID-19: This section shows how the logistic growth model can be useful in predicting the spread of a disease like COVID-19.
- The Basic SIR Model: Introduces the SIR model, a system of three differential equations used to track the populations of infected, recovered, and susceptible individuals in an epidemic.
- Cholera in Haiti: Uses the SIR model to track the spread of Cholera in Haiti during an outbreak.
- CWS Model of Alzheimer's Disease: Introduces the CWS model, a tool that can be useful in tracking the progression of Alzheimer's Disease.
- Gravity Fed Water Delivery: Analyzes an application of Newton's 2nd law of motion (F=ma) on gravity-fed water delivery systems in the form of Bernoulli's equation.
- Earthquake Resistant Construction: Introduces the spring mass equation as it relates to structural dynamics, along with topics like resonance, damping, and forcing.

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
```

(continues on next page)

(continued from previous page)

```
import numpy as np
from scipy.integrate import solve_ivp
```

12.2 Logistic Growth and COVID-19

12.2.1 Introduction

In this section, we explain the difference between exponential and logistic growth models using COVID-19 data reported by Wang et. al. in their 2020 paper "Prediction of epidemic trends in COVID-19 with logistic model and machine learning technics."

12.2.2 Exponential Growth

Let $y(t)$ represent the size of a specified population with $y(0) = y_0$ the initial population size at time $t = 0$. The **exponential growth model** with positive, constant, growth rate k

$$\frac{dy}{dt} = ky, \quad y(0) = y_0$$

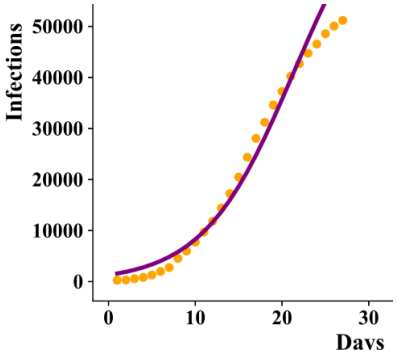
is easily solved by separation of variables:

$$\int \frac{dy}{y} = \int k\,dt \Rightarrow \ln y = kt + C \Rightarrow y = C_1 e^{kt}.$$

Using the initial condition $y(0) = y_0$, we obtain

$$y(t) = y_0 e^{kt}.$$

This model is able to predict rapid growth such as shown in the figure below:



Indeed, if we specify for example $y(10) = 5000$ and $y(20) = 30,000$, we have

$$y(10) = y_0 e^{10k} = 5000 \Rightarrow y_0 = 5000e^{-10k}$$

$$y(20) = y_0 e^{20k} = 30,000 \Rightarrow y_0 = 30000e^{-20k}.$$

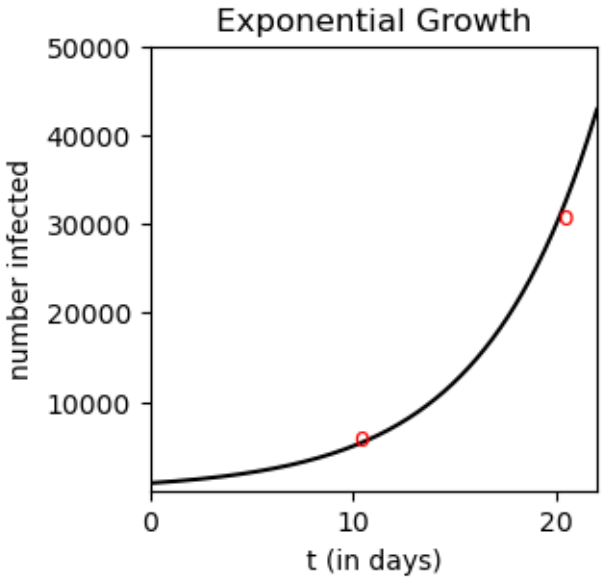
Setting these two expressions for y_0 equal to each other gives

$$5000e^{-10k} = 30000e^{-20k} \Rightarrow e^{10k} = 6 \Rightarrow 10k = \ln 6 \Rightarrow k = \frac{\ln(6)}{10}.$$

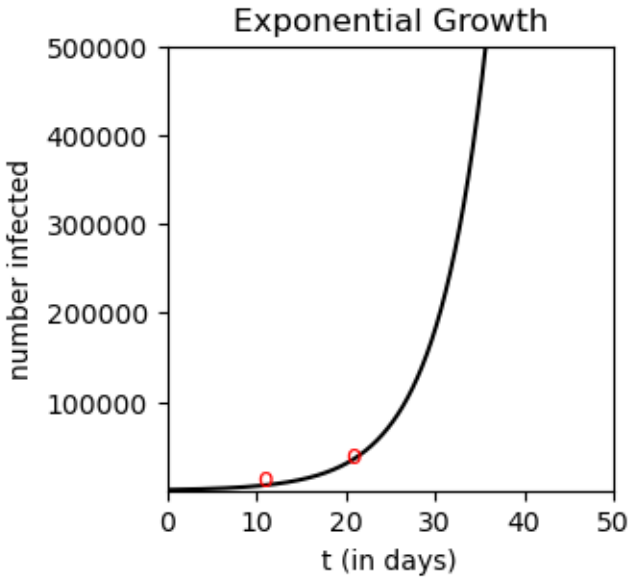
Using this value for k, we can now find y_0:

$$y_0 = 5000e^{-10k} \rightarrow y_0 = 5000e^{-10\frac{\ln(6)}{10}} = 5000e^{-\ln 6} = 5000(\frac{1}{6}) = 5000/6.$$

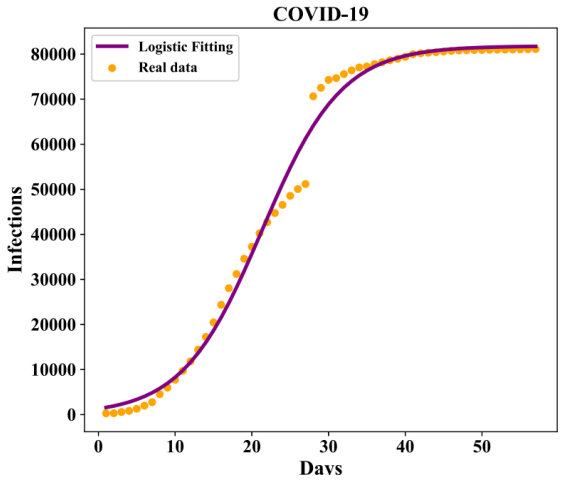
Let's look at a graph of $y = y_0e^{kt}$ for these values of k and y_0.



Any exponential growth function with $y_0 > 0$ and $k > 0$ is such that $\lim_{t\to\infty} y(t) = \infty$, and thus is not realistic for predicting the spread of COVID-19 if the time horizon is sufficiently long. Observe in our case what happens if we extend, for example, the time horizon from $t = 0$ to $t = 50$.



The actual reported data is shown below, and shows $y(t)$ leveling off at about 80000.



12.2.3 Logistic Growth Model

The **logistic growth model**

$$\frac{dy}{dt} = ky(1 - \frac{y}{M}), \quad y(0) = y_0$$

is used to describe "s-shaped" or "sigmoidal" growth. Note that when y is small compared to M, then

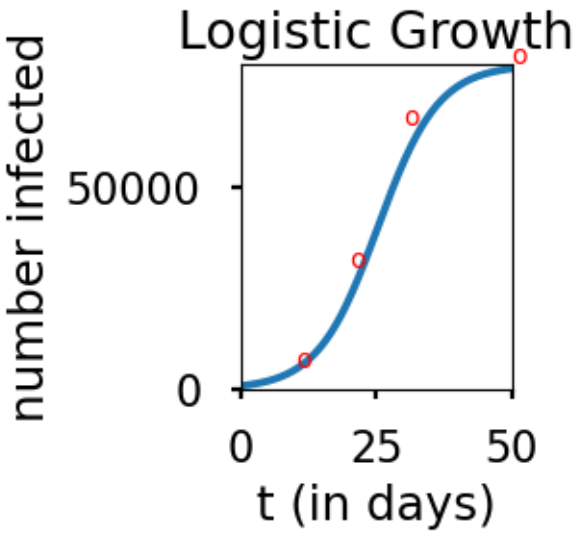
$$\frac{dy}{dt} \approx ky$$

so we he have exponential growth.

However, as y approaches M, the factor $(1 - \frac{y}{M})$ approaches 0, and so $\frac{dy}{dt} \to 0$.This achieves the "leveling off" of the growth.

Let's numerically solve the logistic growth model for the values $k = \log(6)/10$, $y_0 = 5000/6$ and $M = 80000$.

```
C:\Users\pisihara\AppData\Local\Temp\ipykernel_22672\137539061.py:1:␣
↪MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are␣
↪deprecated since 3.6, as they no longer correspond to the styles shipped by␣
↪seaborn. However, they will remain available as 'seaborn-v0_8-<style>'.␣
↪Alternatively, directly use the seaborn API instead.
  plt.style.use('seaborn-poster')
```



12.2.4 Exercises

Exercises

1) Use separation of variables and partial fractions to obtain the general solution to the logistic model

$$\frac{dy}{dt} = ky(1 - \frac{y}{M}) \;\; y(0) = y_0$$

2) a) Make a plot of the logistic model solutions for the values $k = 1$, $M = 100$ for $y_0 = 10$, $y_0 = 50$, $y_0 = 90$, and $y_0 = 110$.

b) Sketch a slope field to explain qualitatively the behavior of the general logistic model for $k > 0$ and $M > 0$. Sketch solution curves where a) y_0 is close to 0; b) y_0 is roughly $M/2$; c) y_0 is slightly smaller than M; and d) y_0 is slightly greater than M

Reference

Wang, P., Zheng, X., Li, J., and Zhu, B. 2020. Prediction of epidemic trends in COVID-19 with logistic model and machine learning technics. Chaos, Solitons, and Fractals, 139(2020), 1-7.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint
```

12.3 The Basic SIR Model

12.3.1 Introduction

The SIR model is a system of three ordinary differential equations that describe the dynamics over time t of the number of susceptible, number of infected, and number of recovered individuals during an epidemic.

Basic SIR Model

$$\begin{aligned} \frac{dS}{dt} &= -\beta SI \\ \frac{dI}{dt} &= \beta SI - \gamma I \\ \frac{dR}{dt} &= \gamma I \end{aligned}$$

where

$S(t)$ the number of susceptible individuals,

$I(t)$ is the number of infected individuals,

$R(t)$ is number of recovered individuals.

The parameter $\beta > 0$ is called the transmission rate, and the parameter $\gamma > 0$ is the recovery rate. (See Fig. 12.1.)

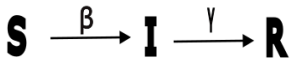


Fig. 12.1: Schematic diagram of transitions.

The differential equations are

Change in Susceptibles Equation (12.1)

$$dS/dt = -\beta SI \tag{12.1}$$

The number of susceptibles (S) decreases with time at a rate proportional to the number of encounters SI between susceptibles and infected (with a constant of proportionality β).

Change in Infected Equation (12.2)

$$dI/dt = \beta SI - \gamma I \tag{12.2}$$

The number of infected I is increasing with respect to time t, proportional to the number of encounters SI between susceptibles and infected (with a constant of proportionality β) and decreasing proportional to the number of infected I present (with a constant of proportionality γ)

Change in Recovered Equation (12.3)

$$dR/dt = \gamma I. \tag{12.3}$$

The number of recovered R is increasing at a rate proportional to the number of infected I (with a constant of proportionality γ.)

12.3.2 Numerical Solutions

We now show how to use an ODE solver to obtain graphs of the solutions $S(t)$, $I(t)$ and $R(t)$ for the following parameter values and initial conditions:

PARAMETER VALUES

β (transmission_rate) = 0.000001

γ (recovery_rate) = 0.2.

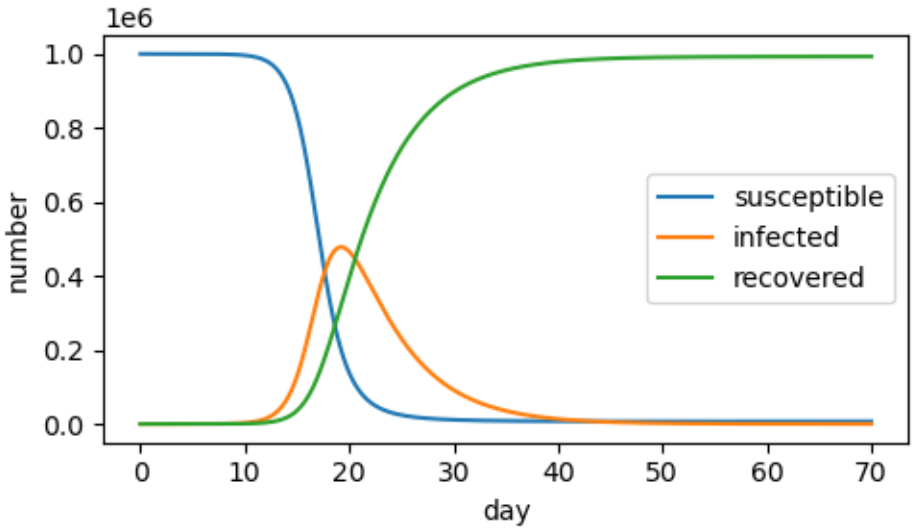
INITIAL CONDITIONS

$S_0 = 1,000,000$ (Initial number of susceptibles)

$I_0 = 1$ (Initial number of infected)

$R_0 = 0$ (initial number of recovered)

```
# Define Equations
def f(y, t, params):
    S, I, R = y      # unpack current values of y
    beta, gamma      # unpack parameters
    #specify dS/dt, dI/dt and dR/dt
    derivs = [-beta * S * I,
              beta * S * I - gamma * I,
              gamma * I]
    return derivs

# Parameters
beta=.000001
gamma=.2
# Initial values
S0=1000000
I0=1
R0=0

# Bundle parameters for ODE solver
params = [beta,gamma]
# Bundle initial conditions for ODE solver
y0 = [S0,I0,R0]

# Make time array for solution
tStop = 70  # number of days
tInc = .001
t = np.arange(0., tStop, tInc)

# Call the ODE solver
psoln = odeint(f, y0, t, args=(params,))

# Plot results
fig = plt.figure(1, figsize=(5,3))
plt.plot(t, psoln[:,0],label='susceptible')
plt.plot(t, psoln[:,1],label='infected')
plt.plot(t, psoln[:,2],label='recovered')
plt.legend()
plt.xlabel('day')
plt.ylabel('number')
plt.savefig("SIR.png")
plt.tight_layout()
plt.show()
```



Note that the transmission rate is sufficiently low that allows for a full recovery of the population (end of epidemic) by roughly day 40.

12.3.3 Exercises

Exercises

1) Investigate how the parameter change affects the disease progression.

(a) What happens if you change the population to a smaller or a larger number? For instance, if you change the population (initial S) to 3,000,000, will the epidemic be over sooner or later? What happens if the population is 1000?

(b) What happens if you change the transmission rate? For instance, if the transmission rate is high, e.g. 0.0001, what do you observe from the curves in the epidemic model? What happens if the transmission rate is lower, e.g. 10^{-7} ?

(c) Does the recovery rate play a role in the epidemic model? If so, how does the recovery rate contribute to the disease progression? Show your adjustments of the recovery rate and their related results.

(d) According to your observations, do you think all people can potentially be infected by the disease? Explain your answer.

(e) Is it possible, based on some initial conditions and parameter values, that there is no disease outbreak? Explain your answer.

2) Consider a modification of the SIR model called the SIRV model for the mathematical modeling of epidemics, where V is the number of vaccinated individuals, and the parameter v>0 is the given constant vaccination rate. The goal is to solve for S(t), I(t), R(t), and V(t). Model and simulate the populations levels S, I, R, and V and interpret the results for this model as above.

In the model, change the equation with dS/dt to convey that not only does the number of susceptibles decrease proportionally to the number of encounters S*I between susceptibles and infected (with a constant of proportionality β), but it also decreases proportionally to the number of vaccinated individuals V present (with a constant of proportionality v). Also, add an equation to convey that the number of vaccinated individuals increases proportionally to the number of susceptibles present (with a constant of proportionality v). Draw a schematic diagram to summarize the movement of people between the compartments S, I, R, and V for the population in this model.

3) Consider a modification of the SIR model called the SIRD model for the mathematical modeling of epidemics, where D is the number of individuals who have died, and the parameter $\mu > 0$ is the given constant mortality rate. The goal is to solve for S(t), I(t), R(t), and D(t). Our new model will simulate the populations levels S, I, R, and D. Visualize and interpret the results for this model as above.

In the SIR model, change the equation with dI/dt to convey that not only does the number of infected increase proportionally to the number of encounters S*I between susceptibles and infected (with a constant of proportionality β) and decrease proportionally to the number of infected individuals I present (with a constant of proportionality γ), but the number of infectives also decreases proportionally to the number of infectives I present (with a constant of proportionality of the mortality rate μ). Also, add an equation to convey that the number of individuals who have died increases proportionally to the number of infected (with a constant of proportionality μ). Draw a schematic diagram to summarize the movement of people between the compartments S, I, R, and V for the population in this model.

12.4 Cholera in Haiti

12.4.1 Introduction

On January 12, 2010, Haiti was struck by a 7.1 magnitude earthquake; this was the largest earthquake this area had seen in more than 200 years. The epicenter was seventeen kilometers southwest of the capital city, Port-au-Prince. Millions of people were affected, with over 200,000 people killed and over 500,000 left homeless. Haiti's woes were exacerbated in October 2010 with one of the worst Cholera outbreaks in history resulting in over 600,000 cases including 8,000 deaths by June 2013. Health officials and scientists are concerned that a reservoir of the Cholera bacteria in Haiti's coastal waters could lead to endemic outbreaks depending on environmental factors such as precipitation and surface water temperature. Consequently, models that consider not only human-to-human transmission processes but also environment-to-human transmission are vitally important. The importance of the Cholera aquatic reservoir was the focus of research in Brazil published 10 years earlier [1], and subsequently revised in the context of Haiti [3]. Important humanitarian goals of cholera modeling include prediction of future outbreaks and reduction of the Cholera bacteria populations to a safe level.

MATLAB Script Objective

The SIR (S=susceptible, I=infected, R=recovered) model from epidemiology can be used to analyze a number of aspects of cholera mathematically ([2]) including:

Basic reproduction number (R0): the average number of susceptible people infected by someone with cholera;

Seasonal variation: the changes in characteristics of disease transmission over time;

Inapparent cholera infections: accounting for the people whose infections are not known or reported;

Human mobility: taking into account the movement of individuals who could spread the disease;

Interventional impact: medical treatment, vaccination, and sanitation.

12.4.2 SIR Model

The SIR cholera model that we will study in this lab was introduced in a Brazilian context [1] fifteen years before the earthquake, but became the basis for models related to the ongoing concern for cholera in Haiti [2, 3]. The basic model is a system of 3 ordinary differential equations

Equation (1)

$$\frac{dS}{dt} = n(H - S) - a\frac{B}{K + B}S$$

Equation (2)

$$\frac{dI}{dt} = a\frac{B}{K+B}S - rI$$

Equation (3)

$$\frac{dB}{dt} = B(nb - mb) + eI,$$

where B(t) is the average reservoir concentration of cholera bacteria (cells/ml), and K is the bacteria concentration giving a 50% chance that a susceptible person catches cholera.

The initial conditions and a set of hypothetical parameters values [1] are given in Table 1.

Table 1

Variable	Meaning	Unit	Initial Value
S	Number of susceptible people	-	1000
I	Number of infected people	-	100
B	Cholera bacteria concentration	cells/ml	0

Parameter	Meaning	Unit	Value [1]
H	total number of people in the community	-	10000
n	growth rate of population	day $^{-1}$	.0001
a	rate of exposure to contaminated water	day $^{-1}$	.5
K	concentration of Cholera bacteria giving a 50% chance that a susceptible person catches cholera	cells/ml	1 x 10 ^{6}
r	rate people recover from Cholera	day $^{-1}$	.2
nb	growth rate of Cholera bacteria in aquatic environment	day $^{-1}$	-
mb	loss rate of Cholera bacteria in aquatic environment	day $^{-1}$	-
$N_b = nb - mb$	(used in MATLAB scripts)	day $^{-1}$	-.33
e	contribution in infected person to Cholera reservoir	$\frac{\text{cell}}{\text{ml day person}}$	10

The system of equations can be easily understood from Fig. 12.2

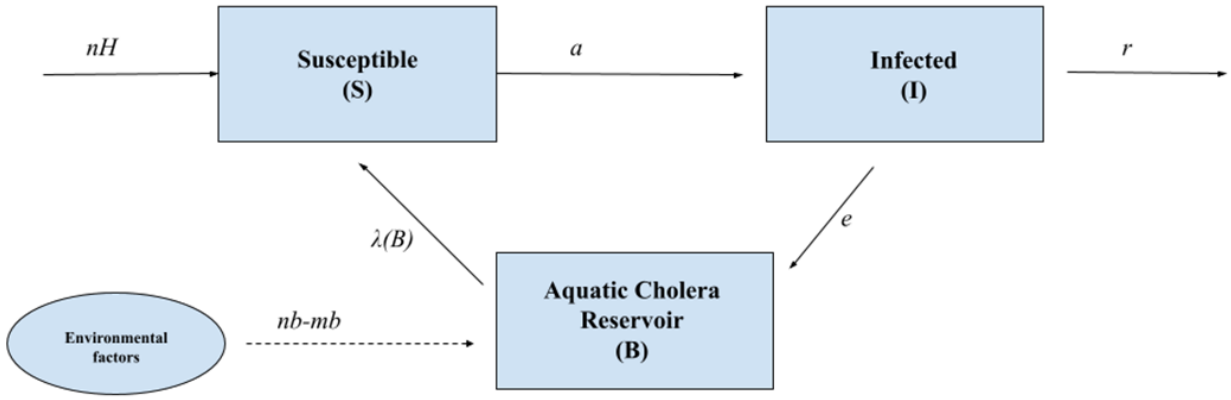


Fig. 12.2: Schematic diagram of transitions. (Adapted from [1]).

12.4.3 Exercises

Exercises

1) Consider the basic SIR model defined by equations (1)-(3). Perform a sensitivity analysis which shows how the peak infected population I depends on the exposure to contaminated water parameter a. (Keep the other parameter and initial values the same as in Table 1.)

2) A water, sanitation, and hygiene intervention (WASH) replacing latrines with flush toilets reduces the contamination rate e. Perform a sensitivity analysis relating e to the peak infected population using the values a = 40 and a = 80. Simulate over a 10 day period.

3) The primary treatment for cholera is oral rehydration which reduces the mortality rate and hence increases the value of the recovery rate r. Perform a sensitivity analysis relating r to the the peak infected population using the values a = 40 and a = 80 keeping fixed e = 10. Simulate over a 10 day period. Comparing your answer with your answers to problems 1 and 2, what might you recommend as an effective intervention strategy?

4) Consider the case where initially no one is infected (I(0) = 0 and S(0) = 10, 000). Perform a sensitivity analysis showing the combined effect of the contamination rate a and initial B(0) aquatic reserve bacteria concentration. (Use the range of values $.1 \leq a \leq 20$ and $.1 \leq B(0) \leq 100$.) What do you observe?

5) Does the basic SIR model with reservoir B(t) (1)-(3) allow for modeling treatment via vaccination? Explain why or why not.

6) What other factors might need to be considered besides minimization of the maximum infected population?

References

[1] Codeco, Claudia Torres, 2001. "Endemic and epidemic dynamics of cholera: the role of the aquatic reservoir." BMC Infectious Diseases 1:1.

[2] Fung, Isaac C-H, 2014, "Cholera Transmission Dynamic Models for public health practioners." Emerging Themes in Epidemiology, 11(1), 1-11.

[3] Kirpich, Alexander, Wappelmann, Thomas, Yang, Yang, Ali, Afsar, Morris, J. Glenn and Longini, Ira, 2015. "Cholera Transmission in Ouest Department of Haiti: Dynamic Modeling and the Future of the Epidemic." PLOS Neglected Tropical Diseases, October 21, 1-12.

[4] Partners in Health, June 11, 2015. "Haiti Continues to Battle Cholera Outbreak" https://www.pih.org/blog/haiti-continues-to-battle-cholera-outbreaks.

12.5 CWS Model of Alzheimer's Disease

Acknowledgements

This material was published previously in an article in the UMAP Journal titled The CWS Model of Alzheimer's Disease. The UMAP Journal 34(1) (2013) 3-18. Copyright 2021 by COMAP, Inc. All rights reserved. Author: Paul Isihara with Anne Bowen, Isaac Brown, Tim Dennison,Wesley Hanna, and Kyle Williams. Permission to make digital or hard copies of part or all of this work for personal or classroom useis granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice. Abstracting with credit is permitted, but copyrights for components of this work owned by others than COMAP must be honored. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior permission from COMAP.

12.5.1 Introduction

The global fight against Alzheimer's Disease (AD) and the role of Amyloid-beta (Aβ) in early detection continue to be a major research concern. AD is now projected to inflict 75 million people in 2030, with the greatest increase in developing countries (Paraskevaidi et al. 2020). Current brain diagnostic tools such as magnetic resonance imaging (MRI) and positron emission tomography (PET) continue to be invasive and expensive.

One theory of how Alzheimer's becomes progressively worse is based on plaques, harmful build-ups of amyloid-beta (Aβ) proteins in the brain. Aβ is called a biomarker since its concentration levels are an observable change in an AD patient's physiology which might help assess the progression of the disease. A more familiar example of a biomarker is cholesterol level for heart disease [Thiess 2010]. In our case, biomarkers such as Aβ may be useful both for early detection of AD which can allow for timely patient care, and also as a measure of the efficacy of therapeutic treatment. As such, a primary motivation for the CWS model introduced 2 decades ago (Craft, Wein and Selkoe 2002), namely, to analyze the use of Amyloid Beta (Aβ) levels in the plasma as a biomarker to detect levels in the brain, continues to be relevant. The CWS-model signaled a need for caution in using Aβ biomarkers since plasma levels under different treatment designs did not directly correlate to brain levels in numerical analysis of the mathematical model (Isihara et al. 2013). Subsequent modeling work by Andou (2013) helped to bridge the gap between mathematical models and in vivo experiments (Carbonell et al. 2018). A problem encountered in clinical trials is the instability and inaccuracy of Aβ plasma level measurements (Park et al. 2017). Recent clinical interest in AD biomarkers goes beyond Aβ plasma analysis to include proposals for eye, saliva, and urine tests (Auso, Gomez-Vicente, and Esquiva 2020).

In this section we review a few highlights of the CWS model. Before we look into the Alzheimer's model, we will look at a more basic example using the Simple Model of the Great Lakes, determining the flow of pollution in the Great Lakes. Interested readers may visit the Alzheimer's Association web site (www.alz.org) for a more comprehensive survey of current research and patient care.

12.5.2 Simple Model of the Great Lakes

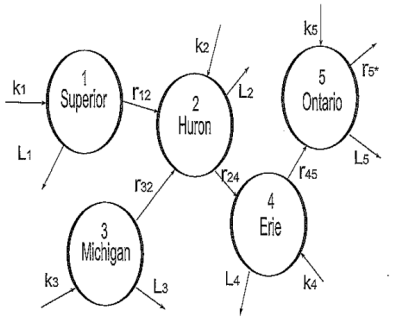
Along the border between the Middle and Eastern portions of the U.S. and Canda, there are five lakes called the Great Lakes. Consider the flow of a pollutant (such as PCB) into, out of, and between the Great Lakes and how one would model the concentration of pollution at time t (see figure below).

We begin by letting $S(t), H(t), M(t), E(t)$ and $O(t)$ represent the respective pollution concentrations (grams/liter) at time t (years) in lake Superior (1), Huron (2),Michigan (3), Erie (4) and Ontario (5). Next we define the following rate parameters:

r_{ij} is the transfer rate (year^{-1}) of pollution from lake i to lake j ;

k_i is the production rate of pollution (grams per liter per year) directly into lake i; and

L_i is the rate of pollution purification treatment (year^{-1}) in lake i.



We make the simplifying assumptions that within each lake (compartment) the pollution concentration is always uniform and the parameters r_{ij}, k_i and L_i are all constant. Under such assumptions, a simple linear system of ODE's can be used to describe the change in pollution concentrations in each lake (compartments):

$$\dot{S}(t) = k_1 - r_{12}S - L_1S = F_1(S, H, M, E, O)$$

$$\dot{H}(t) = k_2 + r_{12}S + r_{32}M - r_{24}H - L_2H = F_2(S, H, M, E, O)$$

$$\dot{M}(t) = k_3 - r_{32}M - L_3M = F_3(S, H, M, E, O)$$

$$\dot{E}(t) = k_4 + r_{24}H - r_{45}E - L_4E = F_4(S, H, M, E, O)$$

$$\dot{O}(t) = k_5 + r_{45}E - r_{5*}O - L_5O = F_5(S, H, M, E, O).$$

Consider, for example, the rate of change at time t of pollution concentration in Huron $\dot{H}(t)$ (in grams/liter per year). This rate of change is equal to: **RateIn** - **RateOut**.

RateIn pollution enters in directly at the rate k_2 (g/L per year) and also enters in from Superior at rate $r_{12}S(t)$ (g/L per year) and from Michigan at rate $r_{32}M(t)$ (g/L per year);

RateOut pollution leaves Huron to Erie at a rate $r_{24}H(t)$ (g/L per year) and also leaves by purification treatment at rate $L_2H(t)$ (g/L per year).

The transfer of pollution between lakes is one-directional as determined by the flow of water.

Standard analysis of this type of model would include:

EQUILIBRIUM ANALYSIS We determine the equilibrium point (s,h,m,e,o) which gives the steady-state concentration levels in each lake by setting each differential equation equal to zero:

$$0 = k_1 - r_{12}s - L_1s$$

$$0 = k_2 + r_{12}s + r_{32}m - r_{24}h - L_2h$$

$$0 = k_3 - r_{32}m - L_3m$$

$$0 = k_4 + r_{24}h - r_{45}e - L_4e$$

$$0 = k_5 + r_{45}e - r_{5*}o - L_5o.$$

Observing that the functions F_i are differentiable with respect to S, H, M, E, O at the equilibrium point, we utilize the Jacobian matrix J which is computed as follows:

$$J = \begin{pmatrix} \frac{\partial F_1}{\partial S} & \frac{\partial F_1}{\partial H} & \frac{\partial F_1}{\partial M} & \frac{\partial F_1}{\partial E} & \frac{\partial F_1}{\partial O} \\ \frac{\partial F_2}{\partial S} & \frac{\partial F_2}{\partial H} & \frac{\partial F_2}{\partial M} & \frac{\partial F_2}{\partial E} & \frac{\partial F_2}{\partial O} \\ \frac{\partial F_3}{\partial S} & \frac{\partial F_3}{\partial H} & \frac{\partial F_3}{\partial M} & \frac{\partial F_3}{\partial E} & \frac{\partial F_1}{\partial O} \\ \frac{\partial F_4}{\partial S} & \frac{\partial F_4}{\partial H} & \frac{\partial F_4}{\partial M} & \frac{\partial F_4}{\partial E} & \frac{\partial F_2}{\partial O} \\ \frac{\partial F_5}{\partial S} & \frac{\partial F_5}{\partial H} & \frac{\partial F_5}{\partial M} & \frac{\partial F_5}{\partial E} & \frac{\partial F_2}{\partial O} \end{pmatrix} =$$

$$\begin{pmatrix} -r_{12} - L_1 & 0 & 0 & 0 & 0 \\ r_{12} & -r_{24} - L_2 & r_{32} & 0 & 0 \\ 0 & 0 & r_{32} - L_3 & 0 & 0 \\ 0 & r_{24} & 0 & -r_{45} - L_4 & 0 \\ 0 & 0 & 0 & r_{45} & -r_{5*} - L_5 \end{pmatrix}$$

The eigenvalues of J determine the stability of the equilibrium point (s, h, m, e, o). In particular, if all the eigenvalues are negative, the equilibrium is stable. In this case, as $t \to \infty$ all solutions approach their equilibrium values; that is, $S(t) \to s$, $H(t) \to h$ $M(t) \to m$, $E(t) \to e$ and $O(t) \to o$.

NUMERICAL ANALYSIS Assuming the rate constants can be estimated by direct measurements or other available data (see for example [Schnoor 1996]), we can use a differential equation solver to obtain the desired pollution concentration levels. If the equilibrium is stable, the equilibrium levels should appear in graphs of their respective pollution concentrations over sufficiently long time intervals.

SENSITIVITY ANALYSIS By varying of one or more of the rate parameters, we can assess the effectiveness of methods to reduce the amount of pollution in a particular lake or the total pollution concentration in all the lakes. For example, since $s = \frac{k_1}{L_1 + r_{12}}$, one can lower the equilibrium concentration in Superior by reducing k_1 (regulating direct contamination), by increasing L_1 (improving purification treatment), and/or by increasing r_{12} (increasing the outflow into Huron).

12.5.3 The CWS Model

Now we will examine the CWS model in relation to Alzheimer's. Craft, Wein and Selkoe [2002] created a model for the amyloid-beta biomarker concentrations in the brain, cerebral spinal fluid ("CSF"), and plasma. Clinically, changes in the concentrations of Aβ in the plasma and CSF are much easier to measure than in the brain. Changes in the CSF and plasma may then be useful to estimate the change in $A\beta$ concentration level in the brain. The CWS model helps to determine the relationship between changes in these three compartments under different treatment scenarios.

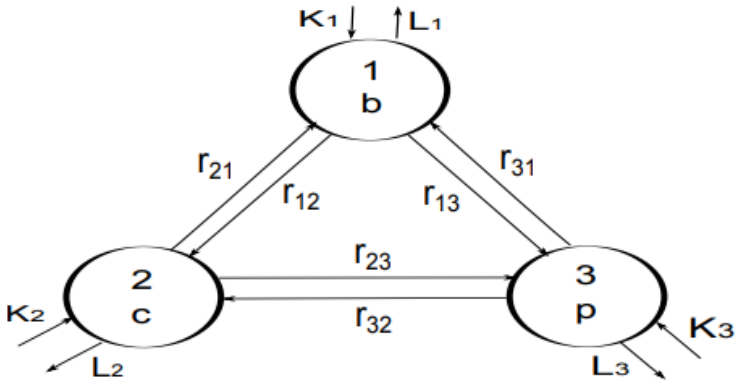


Figure 1: CWS model.

The CWS model can be viewed using compartments similar to the Great Lakes model, as shown in Figure 1. This model is based on several simplifying assumptions:

- Aβ can exist as a single molecule (monomer) or as a chain of molecules (i.e. as a polymer) which connects i monomers ($i = 2, 3, ...$)
- Aβ monomers exist in one of three disjoint compartments: the brain (1), cerebral-spinal fluid or CSF (2) and plasma (3)
- Aβ polymers of any length can exist in the brain (polymer concentrations are assumed to be negligible outside the brain). Note: In **Figure 1** compartment 4 is used to represent the polymers. This ``meta compartment" actually represents an infinite number of compartments, one for each polymer of length i, ($i = 2, 3, ...$)
- Each compartment has a homogeneous concentration of Aβ
- In compartments 1, 2, and 3, Aβ monomers can be created or degraded (lost)
- Monomers can be elongated by adjoining themselves to another monomer or to a polymer of any length
- A polymer of length i can be elongated to a polymer of length $i + 1$
- A polymer of length i can be fragmented into a monomer and a polymer of length $i - 1$. (For more about polymer dynamics, see [Edelstein-Keshet, 2006]).

Under these assumptions, we let $b_1(t)$ denote the Aβ monomer concentration in the brain, $c(t)$ denote the Aβ monomer concentration in the CSF, $p(t)$ denote the Aβ monomer concentration in the plasma, and $b_i(t)$ represent the concentration of length i Aβ polymers in the brain. The following system of ordinary differential equations models the rate of change of the Aβ concentrations in each of the compartments:

(1)

$$\underbrace{\dot{b_1}(t)}_{monomer} = \underbrace{k_1}_{production} + \underbrace{F\sum_{i=2}^{\infty} b_i(t)}_{fragment} - \underbrace{Eb_1(t)(2b_1(t) + \sum_{i=2}^{\infty} b_i(t))}_{all\ elongations} + \underbrace{r_{31}p(t) + r_{21}c(t)}_{monomers\ from\ plasma\ and\ CSF} - \underbrace{(r_{13}b_1(t) + r_{12}b_1(t))}_{monomers\ to\ plasma\ and\ CSF} - \underbrace{L_1b_1}_{los}$$

(2)

$$\underbrace{\dot{b_i}(t)}_{brain\_A\beta(i)-mers} = \underbrace{Eb_1(t)b_{i-1}(t)}_{elongation} + \underbrace{Fb_{i+1}(t)}_{\text{fragmentation of (i+1)-mers}} - \underbrace{Eb_1(t)b_i(t)}_{elongation\ of\ (i)-mers} - \underbrace{F(1 - \frac{1}{2}I_{i=2})b_i(t)}_{fragmentations}$$

(3)

$$\underbrace{\dot{c}(t)}_{\text{CSF } A\beta} = \underbrace{k_2}_{production} + \underbrace{r_{12}b_1(t) + r_{32}p(t)}_{monomers\ from\ brain\ and\ plasma} - \underbrace{(r_{21}c(t) + r_{23}c(t))}_{monomers\ to\ brain\ and\ plasma} - \underbrace{L_2c(t)}_{loss}$$

(4)

$$\underbrace{\dot{p}(t)}_{plasma\ A\beta} = \underbrace{k_3}_{production} + \underbrace{r_{13}b_1(t) + r_{23}c(t)}_{monomers\ from\ brain\ and\ CSF} - \underbrace{(r_{31}p(t) + r_{32}p(t))}_{monomers\ to\ brain and\ CSF} - \underbrace{L_3p(t)}_{loss}.$$

Craft, Wein and Selkoe [2002] describe their model in the following way:

- In Equation (1), the Aβ monomers in the brain are produced by cleavage of the Amyloid precursor protein (APP) at a constant rate of k_1. The monomers live for L_1^{-1} time units before being lost. The same kind of production and loss occur in equations (3) and (4).
- In Equations (1) and (2) we assume that the elongation occurs only by a monomer addition to a polymer with a rate constant E. That is, $Eb_1(t)b_{i-1}(t)$ is the rate $(i-1)$-mers elongate to i-mers.
- Fragmentation occurs when a monomer breaks off at a rate F (i.e., i-mers fragment into an $(i-1)$-mer and a monomer at rate $Fb_i(t)$).
- The factor of 2 in front of $b_1(t)$ in equation (1) arises because it takes two monomers to form a dimer (polymer with length i=2).
- The indicator variable $I_{i=2}$ equals 1 if $i = 2$ and equals 0 otherwise; the factor $\frac{1}{2}$ in front of $I_i = 2$ is due to the fact that a dimer can only fragment in one location, whereas longer i-mers have two possible fragmentation sites.

Equilibrium Analysis

The CWS model is an infinite system of ODEs since it describes changes in the concentrations of polymers of length $i = 2, 3, ...$ Finding the equilibrium solutions of an infinite system is non-trivial, however it "leads to messy solutions and no insight into the problem" [Craft et. al. 2002, p.6]. So, Craft, Wein and Selkoe omit the straightforward approach and present a more intuitive method of finding the equilibrium solutions based on queueing networks [Jackson, 1957]. This more intuitive method is shown below:

Table 1.
Parameter values. [CWS]

Parameter	Description (rate)	Value ($\times$ 86400 day^{-1}) M = molar	Experimental evidence * = computable from estimated steady-state values
k_1	brain monomer production	7.34×10^{-12} M	*
L_1	brain monomer loss	2.78×10^{-4}	mice [Felsenstein, 2000]
E	elongation	90 M^{-1}	synthetic fibril analysis [Lomakin et al. 1997]
F	fragmentation	2.76×10^{-6}	*
k_2	CSF monomer production	0	no contrary data
L_2	CSF monomer loss	0	no contrary data
k_3	plasma monomer production	0	no contrary data
L_3	plasma monomer loss	6.7×10^{-3}	*
r_{12}	brain to CSF monomer transfer	7.6×10^{-6}	mice [Calhoun et al. 1999]
r_{23}	CSF to plasma transfer rate	1.7×10^{-3}	rats [Ghersi-Egea et al. 1996]
r_{31}	plasma to brain monomer transfer	3.77×10^{-5}	guinea pigs [Zlokovic et al. 1993]
r_{32}	plasma to CSF transfer rate	0	[Betz et al. 1989]
r_{21}, r_{13} ,	other transport rates	0	slowness of passive diffusion
Steady-state value	**Description**	**Value ($\times 10^{-12}$ M)**	**Experimental evidence**
b	total brain Aβ_{42}	1975×10^3	dementia patients [Näslund et al. 2000]
$b_1 = 0.013\, b$	brain monomers	25.7×10^3	[McClean et. al. 1999]
c	CSF monomers	115	[Galasko et al. 1998]
p	plasma monomers	29	[Scheuner et al. 1996]

These levels are determined by setting each of the differential equations equal to zero in the CWS model (Equations (1)-(4)):

(1')

$$0 = k_1 + F(b_2 + b_3 + b_4 + ...) - Eb_1(2b_1 + b_2 + b_3 + b_4 + ...) + r_{31}p + r_{21}c - (r_{13}b_1 + r_{12}b_1) - L_1b_1$$

(2') when $i = 2$

$$0 = Eb_1^2 + Fb_3 - Eb_1b_2 - \frac{1}{2}Fb_2$$

(3')

$$0 = k_2 + r_{12}b_1 + r_{32}p - (r_{21}c + r_{23}c) - L_2c$$

(4')

$$0 = k_3 + r_{13}b_1 + r_{23}c - (r_{31}p + r_{32}p) - L_3p.$$

These equations can be solved as follows:

Step 1: Use equations (3') and (4') to solve for c and p in terms of b_1 and constants r_{ij}, k_2, k_3, L_2, and L_3.

In matrix form, equations (3') and (4') become:

(5)

$$\begin{pmatrix} h_2 & -r_{32} \\ -r_{23} & h_3 \end{pmatrix} \begin{pmatrix} c \\ p \end{pmatrix} = \begin{pmatrix} k_2 + r_{12}b_1 \\ k_3 + r_{13}b_1 \end{pmatrix},$$

where $h_2 = r_{21} + r_{23} + L_2$ and $h_3 = r_{31} + r_{32} + L_3$.

Letting $\Delta = h_2 h_3 - r_{23} r_{32}$, the solution to (5) is

$$c = \frac{h_3 k_2 + r_{32} k_3}{\Delta} + \frac{r_{12} h_3 + r_{32} r_{13}}{\Delta}\, b_1 = Q_1 + Q_2 b_1$$

$$p = \frac{r_{23} k_2 + h_2 k_3}{\Delta} + \frac{r_{12} r_{23} + r_{13} h_2}{\Delta}\, b_1 = Q_3 + Q_4 b_1.$$

Step 2: Use equation (2') to express $b_2, b_3, b_4, ...$ in terms of b_1.

Adding the equations when $i = 1$, $i = 2$, $i = 3$, and $i = 4$ together leaves only $Eb_1^2 - \frac{1}{2} F b_2 = 0$, or $b_2 = 2b_1 r$ where $r = b_1 \frac{E}{F}$.

Substituting this expression for b_2 into equation (2'), we obtain $b_3 = 2b_1 r^2$. In a similar way, we can show $b_i = 2b_1 r^{i-1}$, for $i = 2, 3, ...$

Hence, once we find b_1, we can compute $b_2, b_3, ...,$ c and p.

Step 3: Substitute $b_i = 2b_1 r^{i-1}$ into equation (1') to obtain b_1.

$$0 = k_1 + F(2b_1 r^1 + 2b_1 r^2 + ...) - Eb_1(2b_1 + 2b_1 r^1 + 2b_1 r^2 + ...) + r_{31}(Q_3 + Q_4 b_1) + r_{21}(Q_1 + Q_2 b_1) - (r_{13} + r_{12} + L_1) b_1 \Rightarrow$$

$$0 = k_1 + \frac{F 2b_1 r}{1 - r} - \frac{2Eb_1^2}{1 - r} + (r_{31} Q_3 + r_{21} Q_1) + (r_{31} Q_4 + r_{21} Q_2 - r_{13} - r_{12} - L_1) b_1 \Rightarrow$$

$$b_1 = \frac{k_1 + r_{31} Q_3 + r_{21} Q_1}{-r_{31} Q_4 - r_{21} Q_2 + r_{13} + r_{12} + L_1}.$$

This approach indicates:

- The ratio $r = b_1 \frac{E}{F}$ is called the *polymerization ratio*. This a key parameter since all the polymer concentrations can be expressed in terms of r (in particular, $b_i = 2b_1 r^{i-1}$) and further, the total number of polymers forms a geometric series which is convergent if $0 \leq r < 1$ and divergent if $r \geq 1$;
- The steady state monomer concentration b_1 is independent of the polymerization ratio r; and
- The total number of polymers $b = \sum_{i=2}^{\infty} b_i = 2b_1 \sum_{i=2}^{\infty} r^{i-1} = b_1 \frac{2r}{1-r}$ $(r < 1)$ varies directly with b_1.

Parameter Estimation

Assignment of specific values for the rate constants in a model is important and often challenging since real world phenomena are much more complex than their mathematical models. Moreover, in the case of AD, serious ethical issues and physical restrictions limit the type of experimental data which can be obtained. Table 1 gives the values used in the CWS model (with units standardized) and the source of these parameter values:

Table 1.
Parameter values. [CWS]

Parameter	Description (rate)	Value ($\times$ 86400 day^{-1}) M = molar	Experimental evidence * = computable from estimated steady-state values
k_1	brain monomer production	7.34×10^{-12} M	*
L_1	brain monomer loss	2.78×10^{-4}	mice [Felsenstein, 2000]
E	elongation	90 M^{-1}	synthetic fibril analysis [Lomakin et al. 1997]
F	fragmentation	2.76×10^{-6}	*
k_2	CSF monomer production	0	no contrary data
L_2	CSF monomer loss	0	no contrary data
k_3	plasma monomer production	0	no contrary data
L_3	plasma monomer loss	6.7×10^{-3}	*
r_{12}	brain to CSF monomer transfer	7.6×10^{-6}	mice [Calhoun et al. 1999]
r_{23}	CSF to plasma transfer rate	1.7×10^{-3}	rats [Ghersi-Egea et al. 1996]
r_{31}	plasma to brain monomer transfer	3.77×10^{-5}	guinea pigs [Zlokovic et al. 1993]
r_{32}	plasma to CSF transfer rate	0	[Betz et al. 1989]
r_{21}, r_{13}	other transport rates	0	slowness of passive diffusion
Steady-state value	**Description**	**Value ($\times 10^{-12}$ M)**	**Experimental evidence**
b	total brain Aβ_{42}	1975×10^3	dementia patients [Näslund et al. 2000]
$b_1 = 0.013\, b$	brain monomers	25.7×10^3	[McClean et. al. 1999]
c	CSF monomers	115	[Galasko et al. 1998]
p	plasma monomers	29	[Scheuner et al. 1996]

The 4 parameter values "computable from estimated steady state values," obtained from experimental evidence, is particularly interesting. Craft, Wein and Selkoe provide an elegant approach to obtaining equilibrium values and their relationships to these 4 parameters as outlined in **Table 2**:

Table 2.
CWS parameter and steady-state relationships. [CWS]

h_i = total monomer exit rates from compartment i

$h_1 = r_{12} + r_{13} + L_1, \quad h_2 = r_{21} + r_{23} + L_2, \quad h_3 = r_{31} + r_{32} + L_3$

s_{ij} = Survival probability that a monomer leaving compartment i makes it to compartment j

$s_{12} = \frac{r_{12}}{h_1} + \frac{r_{13}}{h_1} s_{32}, \quad s_{21} = \frac{r_{21}}{h_2} + \frac{r_{23}}{h_2} s_{31}, \quad s_{13} = \frac{r_{13}}{h_1} + \frac{r_{12}}{h_1} s_{23},$

$s_{31} = \frac{r_{31}}{h_3} + \frac{r_{32}}{h_3} s_{21}, \quad s_{23} = \frac{r_{23}}{h_2} + \frac{r_{21}}{h_2} s_{13}, \quad s_{32} = \frac{r_{32}}{h_3} + \frac{r_{31}}{h_3} s_{12}$

Direct computation of survival probabilities from transfer and total exit rates

$s_{21} = \frac{r_{21}h_3 + r_{23}r_{31}}{\Delta_1}, \quad s_{31} = \frac{h_2 r_{31} + r_{21} r_{32}}{\Delta_1}, \quad s_{12} = \frac{h_3 r_{12} + r_{13} r_{32}}{\Delta_2},$

$s_{32} = \frac{r_{12}r_{31} + h_1 r_{32}}{\Delta_2}, \quad s_{13} = \frac{h_2 r_{13} + r_{12} r_{23}}{\Delta_3}, \quad s_{23} = \frac{r_{13}r_{21} + h_1 r_{23}}{\Delta_3}$

where $\Delta_1 = h_2 h_3 - r_{23} r_{32}, \quad \Delta_2 = h_1 h_3 - r_{13} r_{31}, \quad$ and $\quad \Delta_3 = h_1 h_2 - r_{12} r_{21}.$

$\hat{k}_i$ = effective monomer production rate for compartment i

$\hat{k}_1 = k_1 + k_2 s_{21} + k_3 s_{31}, \quad \hat{k}_2 = k_2 + k_1 s_{12} + k_3 s_{32}, \quad \hat{k}_3 = k_3 + k_1 s_{13} + k_2 s_{23}$

$\hat{L}_i$ = effective monomer loss rate for compartment i

$\hat{L}_1 = L_1 + r_{12}(1 - s_{21}) + r_{13}(1 - s_{31}), \quad \hat{L}_2 = L_2 + r_{21}(1 - s_{12}) + r_{23}(1 - s_{32}),$

$\hat{L}_3 = L_3 + r_{31}(1 - s_{13}) + r_{32}(1 - s_{23})$

Steady-state formulas

$b_1 = \frac{\hat{k}_1}{\hat{L}_1}, \quad b_i = 2b_1 r^{i-1}\ (i = 2, 3, 4, \ldots), \quad b = \sum_{i=1}^{\infty} b_i = b_1 \left(\frac{2}{(1-r)^2} - 1 \right),$

$c = \frac{\hat{k}_2}{\hat{L}_2}, \quad$ and $\quad p = \frac{\hat{k}_3}{\hat{L}_3}, \quad$ where $r = \frac{\hat{k}_1 E}{\hat{L}_1 F}$

We explain the equations in Table 2 for compartment $i = 1$, the others being similar:

- The total exit rate of monomers from the brain ($h_1 = r_{12} + r_{13} + L_1$) is the sum of the respective rates monomers move from brain to CSF (r_{12}) and from brain to plasma (r_{13}), these being added to the loss of monomers in the brain by degradation (L_1);
- The survival probability that a monomer exiting the brain reaches the CSF ($s_{12} = \frac{r_{12}}{h_1} + \frac{r_{13}}{h_1} s_{32}$) is the probability

that an exiting brain monomer goes directly to the CSF ($\frac{r_{12}}{h_1}$) plus the probability that a monomer first goes from brain to plasma and then transfers from plasma to CSF ($\frac{r_{13}}{h_1}s_{32}$);

- The effective production of monomers in the brain ($\hat{k}_1 = k_1 + k_2 s_{21} + k_3 s_{31}$) is the sum of direct production (k_1) plus the rate monomers transfer in from the CSF ($k_2 s_{21}$) and from the plasma ($k_3 s_{31}$); and
- The effective loss of monomers in the brain ($\hat{L}_1 = L_1 + r_{12}(1 - s_{21}) + r_{13}(1 - s_{31})$) is due to degradation L_1 and also by transfer out to and non-return from the CSF ($r_{12}(1 - s_{21})$) and by transfer out to and non-return from the plasma ($r_{13}(1 - s_{31})$).

12.5.4 A Simplified CWS Model

The full CWS model (Equations (1)-(4)) does not have an exact solution nor can it be numerically integrated even with assigned parameter values because it is nonlinear and infinite dimensional. Therefore, the infinite system must be simplified in some way to a finite system. One such simplification is to approximate the CWS model monomer dynamics by removing the polymer compartment from the CWS model. In other words, we assume for simplicity that the values of E and F are negligible and set them both equal to zero. We keep all the other parameter values in Table 1 the same. The simplified CWS model based on only monomers being transferred takes the form of an autonomous linear system:

(L1)

$$\underbrace{\dot{b}_1\,(t)}_{monomer} = \underbrace{k_1}_{production} + \underbrace{r_{31}p(t) + r_{21}c(t)}_{monomers\,from\,plasma\,and\,CSF} - \underbrace{(r_{13}b_1(t) + r_{12}b_1(t))}_{monomers\,to\,plasma\,and\,CSF} - \underbrace{L_1 b_1(t)}_{loss}$$

(L2)

$$\underbrace{\dot{c}\,(t)}_{CSF\,A\beta} = \underbrace{k_2}_{production} + \underbrace{r_{12}b_1(t) + r_{32}p(t)}_{monomers\,from\,brain\,and\,plasma} - \underbrace{(r_{21}c(t) + r_{23}c(t))}_{monomers\,to\,brain\,and\,plasma} - \underbrace{L_2 c(t)}_{loss}$$

(L3)

$$\underbrace{\dot{p}\,(t)}_{plasma\,A\,\beta} = \underbrace{k_3}_{production} + \underbrace{r_{13}b_1(t) + r_{23}c(t)}_{monomers\,from\,brain\,and\,CSF} - \underbrace{(r_{31}p(t) + r_{32}p(t))}_{monomers\,to\,brain\,and\,CSF} - \underbrace{L_3 p(t)}_{\text{loss}}.$$

Approximating the monomer dynamics may be useful as a measure of the total amyloid-beta concentration since, as we observed at the end of the last section, the total steady state polymer concentration varies directly with steady state monomer concentration b_1. (Recall that an elevated total $A\beta$ concentration level in the brain is thought to produce senile plaques which may be a major factor contributing to the progression of AD. (See [Luca et. al. 2003] for mathematical modeling related to plaque formation).

Since we use the same parameter values given in Table 1 for the simplified (linear) CWS model, the respective equilibrium values for monomers in the brain ($b_1 \sim 25,700$ pM), CSF ($c \sim 115$ pM) and plasma ($p \sim 29$ pM) remain the same as in Table 1. The stability of these equilibria is confirmed by computing the (constant) Jacobian matrix of the simplified model

$$J = \begin{pmatrix} -r_{13} - r_{12} - L_1 & r_{21} & r_{31} \\ r_{12} & -r_{21} - r_{23} - L_2 & r_{32} \\ r_{13} & r_{23} & -r_{31} - r_{32} - L_3 \end{pmatrix} \approx \begin{pmatrix} -24.6566 & 0 & 3.25728 \\ .65664 & -146.88 & 0 \\ 0 & 146.88 & -582.137 \end{pmatrix},$$

and finding its eigenvalues, {-582.136,-146.886,-24.652}, are all negative at the equilibrium point (b_1, c, p).

Craft, Wein and Selkoe took a different approach to simplifying the infinite dimensional system by assuming that the total polymer concentration remains constant ($\sum_{i=2}^{\infty} b_i(t) = S$ = constant). It is easy to show that their assumption leads to the following 3 dimensional non-linear system for $b_1(t), c(t)$ and $p(t)$:

(NL1)

$$\dot{b}_1\,(t) = \tilde{k}_1 + r_{31}p(t) + r_{21}c(t) - (r_{13}b_1(t) + r_{12}b_1(t)) - \tilde{L}_1 b_1(t) - 2Eb_1^2(t)$$

(NL2)

$$\dot{c}\,(t) = k_2 + r_{12}b_1(t) + r_{32}p(t) - (r_{21}c(t) + r_{23}c(t)) - L_2c(t)$$

(NL3)

$$\dot{p}\,(t) = k_3 + r_{13}b_1(t) + r_{23}c(t) - (r_{31}p(t) + r_{32}p(t)) - L_3p(t)$$

where $\tilde{k}_1 = k_1 + FS$ and $\tilde{L}_1 = L_1 + ES$. Except for the non-linear term $-2eb_1^2$ and adjusted parameter values $\tilde{k}_1$ and $\tilde{L}_1$ in (NL1), this system is identical to the linear system given by (L1).

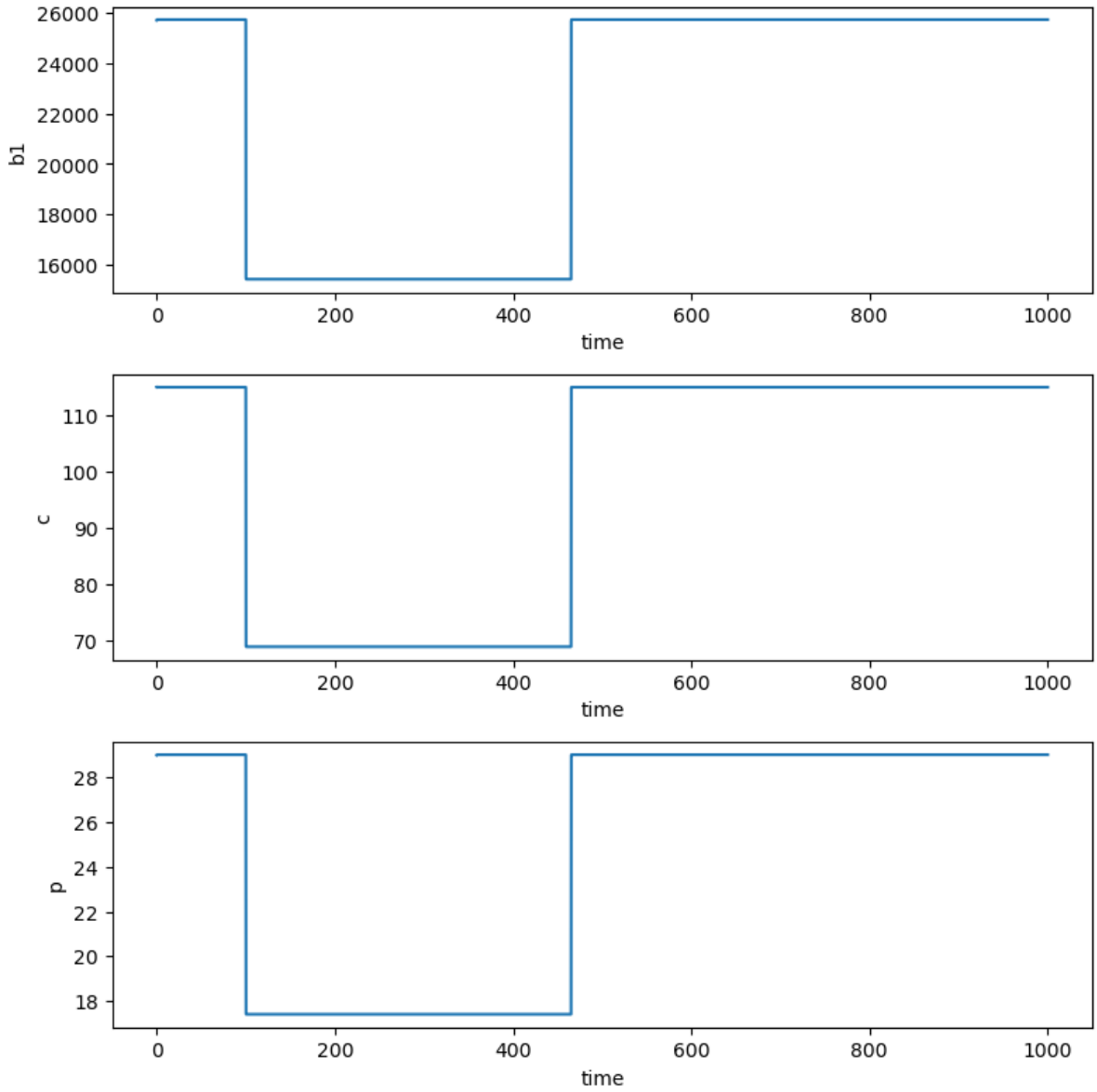
We found that in all cases of interest these two simplified systems exhibited very similar numerical solution behavior (solutions are strongly attracted by the equilibrium values) and so will only graph numerical solutions to the simplified CWS model (L1) - (L3).

12.5.5 Analysis of Treatments

Craft, Wein and Selkoe [2002] simulated a variety of treatment scenarios to assess the suitability of using $A\beta$ monomer concentrations in the CSF and plasma to predict the $A\beta$ monomer concentration in the brain. To illustrate and confirm their findings, we use the simplified CWS model (L1) - (L3) to simulate two of these treatment scenarios.

Treatment 1

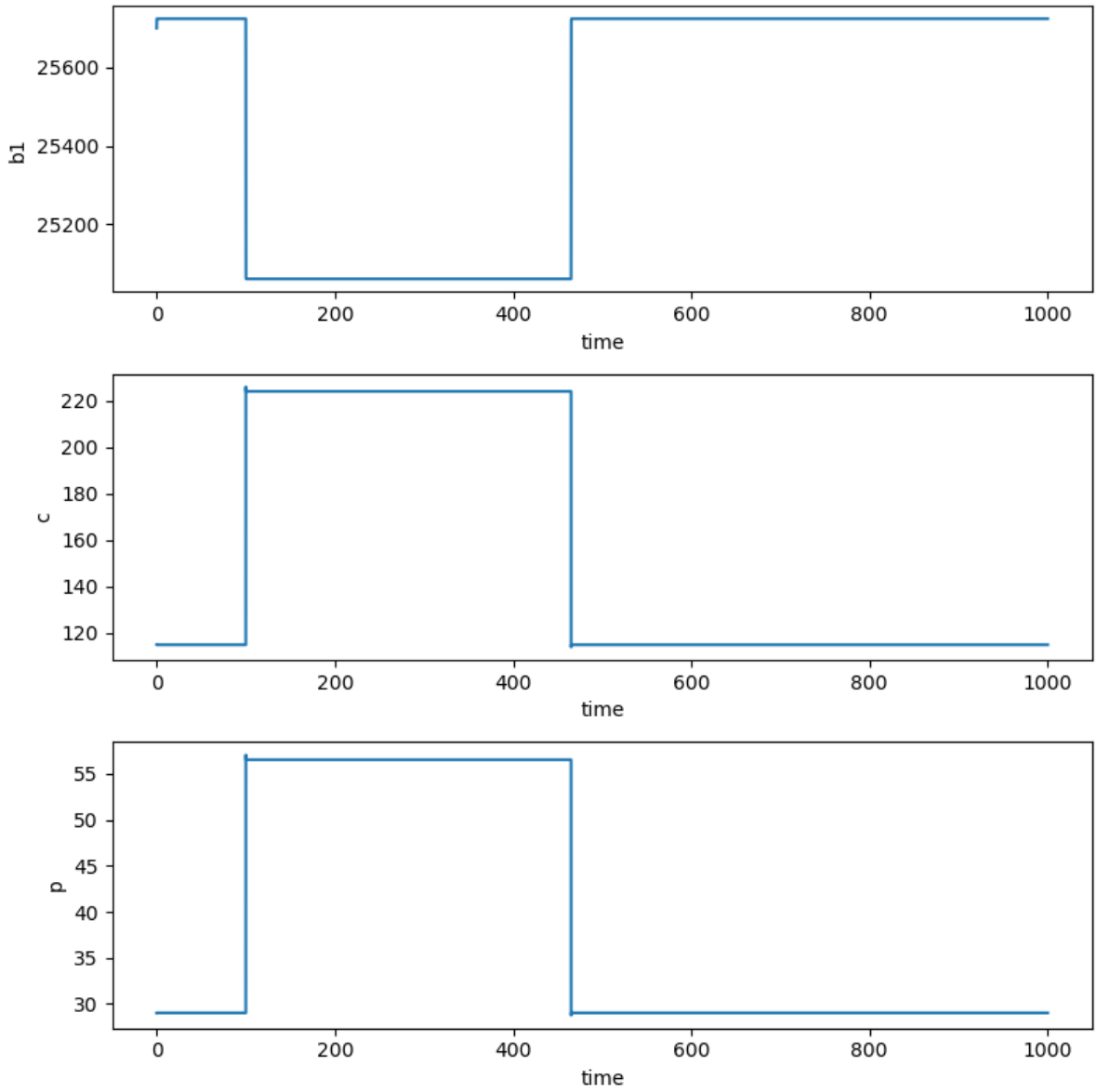
The first treatment scenario is to reduce the production of Aβ in the brain, that is, to lower the value of k_1. For such a treatment, a γ-secretase inhibitor is used [Craft et. al. 2002]. (See [Das et. al. 2011] for further mathematical modelling on γ-secretase dynamics between the CSF and plasma). To simulate the effect of the treatment, we solve Equations L1-L3 numerically. We are assuming that a symptomatic patient is in a pre-treatment steady state on days 0-100 and is given a γ-secretase inhibitor that decreases the production rate k_1 by 40% on days 100-465.



Treatment 1 results in the same Amyloid-beta level changes in the brain, CSF and plasma.

Treatment 2

Our second treatment scenario is to change the transfer rate of Aβ between the brain and CSF. We will investigate what happens when r_{12} is increased by 100% on days 100-465. We simulate the effect of this treatment below. Why does this caution the use of Amyloid-beta as a biomarker for Alzheimer's Disease?



The Aβ concentration in the brain during treatment 2 decreases by 2.5% while the concentrations in the CSF and plasma both roughly double (in agreement with Figure 6 b-d in [Craft et. al. 2002]).

Note: Craft, Wein and Selkoe [2002] considered two other treatments:

- A fragmentation enhancer increasing the fragmentation rate 100%; and
- A vaccine which increases the monomer ingestion rate off of polymers by 100%.

In both these cases, the dynamics in all three compartments were similar.

12.5.6 Conclusion

After simulating a variety of treatment scenarios, Craft, Wein and Selkoe concluded the need for caution about the use of Aβ as a biomarker. As exemplified by Treatment 1 and 2, relative changes in the Aβ levels are not uniform in all three compartments under all treatments. The effects of a particular treatment on the model parameters must be understood in order to correctly interpret clinical changes in a patient's CSF and plasma Aβ levels.

Craft, Wein and Selkoe observe, for the case where the polymerization ratio satisfies $r > 1$, that though the values in Table 1 give $r \sim .84$ and a total steady-state polymer concentration $S \sim 1950\text{x}10^3$ pM, theoretically speaking, if $r > 1$, the steady state polymer concentration S becomes infinite. This implies that while the steady state brain monomer concentration b_1 (which is independent of r) remains finite or is even reduced by some treatment, the total brain polymer concentration $S = \sum_{i=2}^{\infty} b_i$ may increase without bound as each polymer concentration b_i approaches its finite steady state level (see Figure 5 in [Craft et. al. 2002]).

12.5.7 Exercises

Exercises

1) The equilibrium analysis for Equation (2) of the CWS Model shown in this notebook is conducted for $i = 2$. Since i is the length of polymers, it is possible for i to equal 2, 3, 4, …. Using Equation (2) in the CWS Model, write what the equation used for equilibrium analysis of Equation (2) would be when $i = 3$ and $i = 4$.

2) Explain the equation for h_2 in Table 2.

3) Why does the output for Treatment 2 caution the use of Amyloid beta as a biomarker for Alzheimer's disease?

4) Simulate Treatment 2 using the non-linear model given by equations NL1-NL3. (Use the values $E = S = 0$ and $F = .238$.) What do you observe?

12.5.8 References

Calhoun, M., Burgermeister, P., Phinney, A., Stalder, M., Tolnay, M., Wiederhold, K.-H., Abramowski, D., Sturchler-Pierrat, C., Sommer, B., Staufenbiel, M. ad Jucker, M. 1999. Neurol overexpression of mutant amyloid pre-cursor protein results in prominent deposition of cerebrovascular amyloid. *Neurobiology* 96, 14088-14093.

Craft, D., Wein, L. and Selkoe,D. 2002. Mathematical model of the impact of novel treatments of the Aβ burden in the Alzheimer's brain, CSF and plasma, *Bulletin of Mathematical Biology*, 64, 1011-1031.

Das, R. et. al. 2011. Modelling effect of a γ-Secratase inhibitor on amyloid-β dynamics reveals significant role of an amyloid clearance mechanism, *Bulletin of Mathematical Biology*, 73, 230-247.

Edekstein-Keshet, L. 2006. Simple models for polymer growth: implications for biopolymers, available at https://people.ok.ubc.ca/rtyson/Teaching/Math225/polymerize.pdf

Felsenstein, K. 2000. The next generation of AD therapeutics: the future is now. *Abstracts from the 7th annual conference on Alzheimer's Disease and Related Disorders*, Abstract 613.

Galasko, D. et. al. 1998. High cerebrospinal fluid tau and low amyloid β42 levels in the clinical diagnosis of Alzheimer disease and relation to Apolipoprotein E genotype, *Archives of Neurology*, 55, 937-945.

Ghersi-Egea, J.-F., Gorevic, P., Ghiso, J., Frangione, B., Patlak, C., Fenstermacher, J. 1996. Fate of cerebrospinal fluid-borne amyloid β-peptide: rapid clearance into blood and appreciable accumulation by cerebral arteries. *Journal of Neurochemistry*, 166, 880-883.

Jackson, J. 1957. "Networks of waiting lines" *Operations Research*, 5, pp. 518-521.

Johns H. and King L. 2011. "Larry King Interviews Harry Johns, President of the Alzheimer's Association." in the CNN special program *Larry King Investigates Alzheimer's.* Available at https://web.archive.org/web/20140907054847/http://www.alzheimersreadingroom.com/2011/05/harry-johns-larry-king-investigates.html

Luca, M., Chavex-Ross, A, Edelestein-Keshet, L. and Mogilner A. 2003. Chemotactic signaling, microglia, and Alzheimer's disease senile plaques: is there a connection?, *Bulletin of Mathematical Biology*,, 65, 693-730.*

McLean, C., Cherny, R., Fraser, F., Fuller, S., Smith, M., Beyreuther, K., Bush, A., Masters, C. 1999. Soluble pool of Aβ amyloid as a determinant of severity of neurodegeneration in Alzheimer's Disease. *Annals of Neurology*, 46, 860-866.

Näslund, J., Haroutunian, V., Mohs, R., Davis, K., Davies, P., Greengard, P., Buxbaum, J. 2000. Correlation between elevated levels of amyloid β-peptide in the brain and cognitive decline. *The Journal of the American Medical Association}, 283, 1571-1577.*

Scheuner, D. et. al. 1996. *Nature Medicine*, 2, 864-870.

Schnoor, Jerald L. 1996. *Environmental Modeling; Fate and Transport of Pollutants in Water, Air, and Soil*, New York: John Wiley & Sons.

Thiess, W. 2010. Regarding ADNI Biomarkers Article from *Archives of Neurology* August 2010.

Zlokovic, B., Ghiso, J., Mackic, J., McComb, J., Weiss, M. and Frangione, B. 1993. Blood-brain barrier transport of circulating Alzheimer's amyloid β, *Biochemical and Biophysical Research Communications*, 197, 1034-1040.

12.6 Gravity Fed Water Delivery

Newton's 2nd Law of motion $F = ma = m\frac{dv}{dt}$ is a classic differential equation with an amazingly wide range of physical applications. In this section, we consider an application of Newton's 2nd law to analysis of gravity-fed water delivery systems. Understanding how the limit-quotient definition of derivative arises in a physical context is an important idea in understanding how differential equations governing physical processes are derived. In this case, solving the differential equation is simple. Antidifferentiation of the differential equation gives a solution which is called Bernoulli's Law. We will then show how Bernoulli's law applies to practical design of gravity-fed water delivery systems.

Acknowledgements

This material was published previously as a UMAP Module titled Gravity-Fed Water Delivery Systems. The UMAP Journal 31(4) (2010) 281-322. Copyright 2010 by COMAP, Inc. All rights reserved. Authors: Andrew Dornbush and Paul Isihara with Timothy Dennison, Kristianna Russo, and David Schultz. Permission to make digital or hard copies of part or all of this work for personal or classroom useis granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice. Abstracting with credit is permitted, but copyrights for components of this work owned by others than COMAP must be honored. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior permission from COMAP.

12.6.1 Introduction

According to the World Health Organization (WHO), in 2022, around 2 billion people worldwide lived without close proximity to a safe water resource [https://www.who.int/news-room/fact-sheets/detail/drinking-water]. Unsafe water sources and lack of sanitation cause diseases and disabilities such as malaria, diarrhea, and blindness. These diseases kill an estimated 3.5 million people per year with most of the victims under five years old. Furthermore, unprotected water sources, such as surface and ground water, can become polluted by pesticides and by household and industrial waste. Additionally, agricultural and industrial over-extraction threatens the sustainability of these resources.

Globally, there exists a disparity in access to safe water between urban and rural populations. Simple, affordable pipe systems supply spring water to impoverished rural villages that would otherwise have no water delivery. The simplest of these systems are entirely gravity-powered; gravity alone moves water that flows through pipes directly from a source tank

supplied by a natural spring at a higher elevation to a receiving tank for distribution at a lower elevation as shown in Figure 1 below.

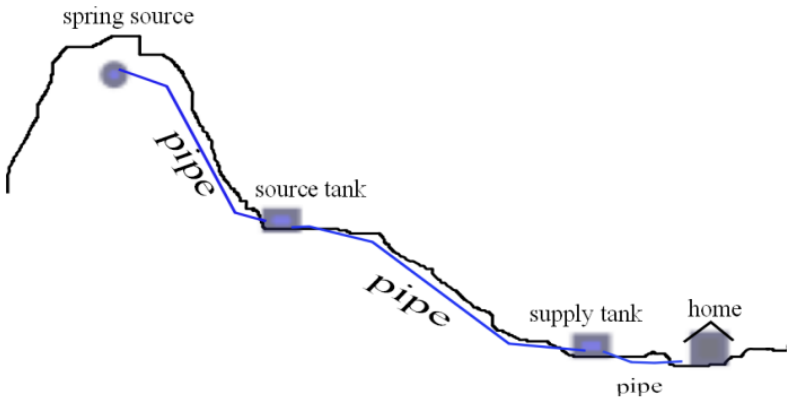


Figure 1. Simple systems use only gravity to transport water.

In this section we will introduce and apply basic principles from fluid mechanics to construct a mathematical model of such gravity-fed water delivery systems. This model provides insight into calculations needed for actual construction (Figure 2), and serves as an excellent example of how even beginning calculus students (concurrently taking physics) can engage in a modeling problem of great practical importance.

Figure 2. Construction of gravity-fed systems utilizes mathematical models.

We begin in Section 1.1 by using elementary physics to derive a basic form of Bernoulli's equation, which describes conservation of energy for a fluid particle flowing along a streamline. Next, in Section 1.2 we apply Bernoulli's equation to a simple gravity-fed water delivery system. We introduce the concept of hydraulic heads to help visualize the conversion of potential energy into fluid pressure, revealing the need for break-pressure tanks to avoid any risk of pipe explosion. We also analyze the conversion of potential to kinetic energy to predict flow rates (important for meeting the client village's demand). Section 1.3 explains how the basic model is modified to account for friction and different kinds of water flow. In particular, we show how the fluid velocity changes for laminar or viscous flow and explain qualitatively the use of Moody charts in finding the friction factor for turbulent flow. Section 1.4 gives an example of steps involved in overall system design. Section 1.5 looks at how basic water delivery systems have been effectively used in Honduras. Finally, Section 1.6 gives suggestions for those interested in further study. Table 1 serves as a hand reference for the notation and units used in this Module.

Table 1.
Notation.

Symbol	Meaning	Metric Units	US Units	Metric Value for Water (at 20° C)
u	Velocity	$\mathrm{m/s}$	$\mathrm{ft/s}$	—
a	Acceleration	$\mathrm{m/s^2}$	$\mathrm{ft/s^2}$	—
F	Force	N	$\mathrm{lb\text{-}ft/s^2}$	—
ρ	Density	$\mathrm{kg/m^3}$	$\mathrm{lb/ft^2}$	998.2
p	Pressure	$\mathrm{N/m^2}$	$\mathrm{lb/ft\text{-}s^2}$	—
Q	Volumetric Flow-rate	$\mathrm{m^3/s}$	$\mathrm{ft^3/s}$	—
γ	Specific Weight	$\mathrm{N/m^3}$	$\mathrm{lb/ft^2\text{-}s^2}$	9789
μ	Dynamic Viscosity	$\mathrm{kg/m\text{-}s}$	$\mathrm{lb/ft\text{-}s}$	1.002×10^{-4}
τ	Shear Stress	$\mathrm{N/m^2}$ ($\perp$ area)	$\mathrm{lb/ft\text{-}s^2}$	—

12.6.2 Derivation of Bernoulli's Equation

Our goal is to model the motion of a water particle along a path called a *streamline* when building a simple system to deliver water through a pipe. *Bernoulli's equation* is fundamental in fluid mechanics for analyzing the motion of a fluid particle (not only liquid) along a streamline and is of particular importance in our analysis of gravity-fed water delivery systems. We will derive a basic form of this equation, following Young, et. al. [2001], using several simplifying assumptions compiled in Table 2.

Table 2.
Assumptions used in deriving a basic form of Bernoulli's equation.

Assumption	**Description**
A1. *Incompressible*	Fluid density $\left(\frac{\text{mass}}{\text{volume}}\right)$ is constant in both space and time.
A2. *Gravity-Fed*	Total system energy comes solely from gravitational potential. The system neither creates nor absorbs force, meaning it has no *sources* (e.g., pumps) or *sinks* (e.g., turbines) of energy.
A3. *Steady State*	Fluid velocity at each point in space does not vary over time.
A4. *Inviscid*	Flow is frictionless; all forces (gravity, pressure) are conservative. (**Section 4** modifies our model to account for friction.)

We start by representing a fluid particle as a differential volume element shaped as a rectangular prism with size δV cubic units, and look at the forces acting on it. To analyze this motion, we introduce the $(\hat{s},\hat{y},\hat{n})$ orthogonal curvilinear coordinate system (see Figure 3). If we assume the streamline is situated in a single plane, then the $(\hat{s},\hat{y},\hat{n})$ coordinate system is obtained from $(\hat{x},\hat{y},\hat{z})$ coordinates by keeping the $\hat{y}$ axis (which is normal to the plane) fixed and rotating the $\hat{x}$ and $\hat{z}$ axes by a negative angle θ to obtain the $\hat{s}$ and $\hat{n}$ axes respectively. The positive $\hat{s}$ axis gives the direction of fluid particle motion along a path called a *streamline*.

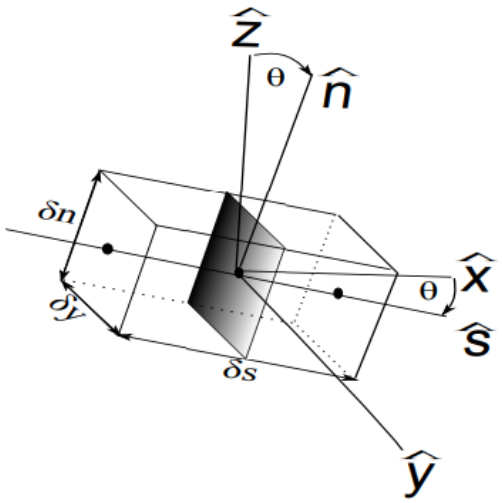


Figure 3. A fluid particle is represented as a differential volume element with size $\delta V = \delta s\, \delta y\, \delta n$.

$\hat{s}$= distance traveled along a streamline (note that the unit vector in the positive $\hat{s}$ direction is always tangent to the streamline);

$\hat{n}$= normal to the streamline; and

$\hat{y}$= perpendicular to the $\hat{s} - \hat{n}$ plane.

At each point along its path of motion, the total force $\vec{F}_{total}$ on the fluid particle (control volume) satisfies Newton's 2nd Law:

Equation (1)

$$\vec{F}_{total} = m\, \vec{a},$$

where m is the mass of the control volume and $\vec{a}$ is its acceleration. Since we assume the system doesn't create or absorb force, meaning it has no sources or sinks of energy (assumption A2) and there is no friction (inviscid flow–assumption A4), the only forces acting on the particle will be due to

gravity: ($\vec{F}_{gravity}$), acts in the downward (-$\hat{z}$) direction; and

fluid pressure: ($\vec{F}_{pressure}$), acts in the $\hat{s}$ direction.

Thus we calculate net force and the components of forces in the direction of the streamline as follows:

$$\vec{F}_{total} = \vec{F}_{pressure} + \vec{F}_{gravity} \quad \Rightarrow$$

Equation (2)

$$F_s = F_{ps} + F_{gs}$$

where (see Figure 4)

F_s is the component of the total force along the streamline;

F_{ps} is the component of the pressure force along the streamline; and

F_{gs} is the component of the gravitational force along the streamline.

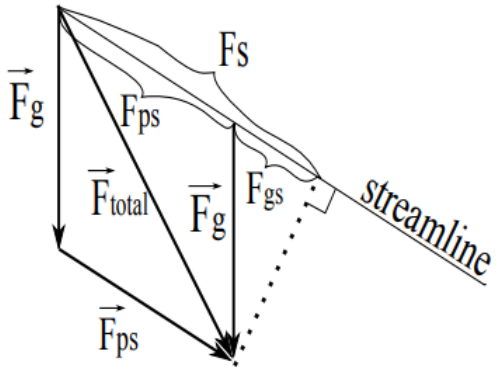


Figure 4. The component of the total force (F_s) on a particle in the direction of a streamline is the sum of its pressure (F_{ps}) and gravity (F_{gs}) components.

Note that the acceleration of the control volume at time t will be $\vec{a}=\vec{a_s}=\frac{d\vec{u}}{dt}$, where $\vec{u}\ (t)$ is the fluid particle's velocity which, by definition, is along the streamline. (For curved water pipes and streamlines, beyond the scope of this Module, the normal component of acceleration (a_n) is considered when computing pressure variation across a streamline [Young et. al. 2001].) Recalling that $u=|\vec{u}|=\frac{ds}{dt}$, by the chain rule of differentiation, the component a_s of the acceleration $\vec{a_s}$ of the control volume in the direction of the streamline is

$$a_s = \frac{du}{dt} =$$

$$\frac{du}{ds}\frac{ds}{dt} =$$

Equation (3)

$$u\frac{du}{ds}.$$

Noting that $F_s = ma_s$, by equations (2) and (3),

$$F_s = mu\frac{du}{ds} \Rightarrow$$

Equation (4)

$$F_{gs} + F_{ps} = \rho\delta V u\frac{du}{ds},$$

where the mass m of the control volume is the product of the mass density ρ (mass per unit volume) and volume δV .

We will now obtain expressions for F_{gs} and F_{ps} in equation (2). First, to find F_{gs} we observe that the magnitude of the force due to gravity is given by

Equation (5)

$$F_g = mg = \rho g\delta V,$$

which we recall is in the $-\hat{z}$ direction. It is convenient to define the *specific weight* $\gamma = \rho g$, so we can rewrite equation (5) as

Equation (6)

$$F_g = \gamma\delta V.$$

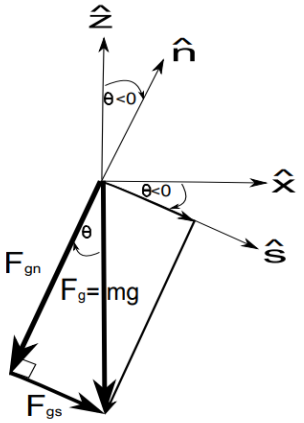


Figure 5. Component F_{gs} of the gravitational force in the $\hat{s}$ direction.

From equations (5) and (6), the component of the gravitational force acting on the control volume along the direction of the streamline will be

$$F_{gs} = -F_g \sin\theta =$$

Equation (7)

$$-\gamma \delta V \sin\theta.$$

Note that the negative sign is needed since $-\frac{\pi}{2} < \theta < 0 \Rightarrow -\sin\theta > 0$.

Our next task is to find the magnitude F_{ps} of the force on the control volume (in the direction of the streamline) due to pressure. For steady, inviscid flow (assumptions A3, A4), pressure acts perpendicular to each point of every cross section and its magnitude is a function of s only; that is, $F_{ps} = p(s)$. Suppose the center of the control volume is at $\hat{s} = s_0$. Recalling that pressure is force per unit area, let $p(s_0)$ be the magnitude of the pressure on the rectangular cross-section $\mathcal{R}_0$ which is perpendicular to the streamline at $\hat{s} = s_0$ (see Figure 6).

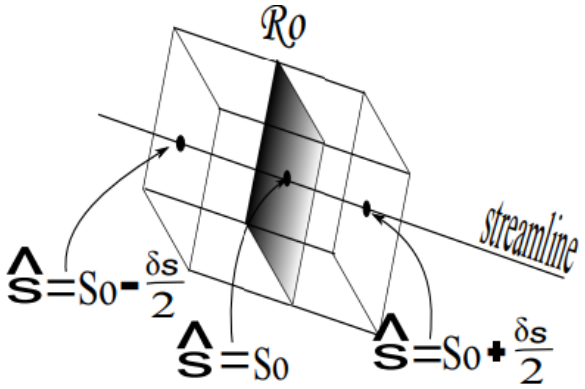


Figure 6. Computation of pressure on the control volume at position $\hat{s} = s_0$.

The magnitude of the pressure at the two ends of the control volume parallel to $\mathcal{R}_0$ are $p_{right} = p(s_o + \frac{\delta s}{2})$ and $p_{left} = p(s_o - \frac{\delta s}{2})$. A linear approximation for p_{right} (Figure (7)) is given by

Equation (8)

$$p_{right} = p(s_o + \frac{\delta s}{2}) = p(s_o) + \delta p_s \approx p(s_o) + \frac{dp}{ds}\frac{\delta s}{2}.$$

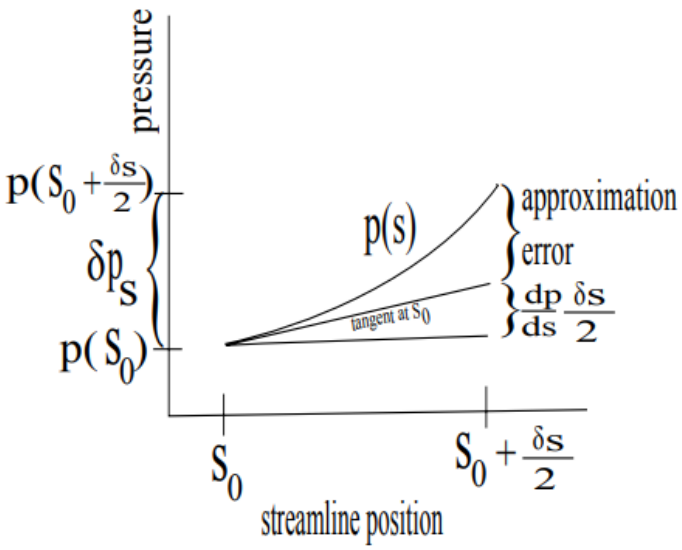


Figure 7. Linear approximation of the change in pressure δp_s for small δs.

As $\delta s \to 0$ the approximation error also approaches 0. The control volume is infinitesimal, so Equation (8) is a good approximation. In a similar way, we approximate p_{left} using $p(s_o - \frac{\delta s}{2}) \approx p(s_0) - \frac{dp}{ds}\frac{\delta s}{2}$.

Since pressure is force per unit area, and the cross-sectional area is $\delta A = \delta n \delta y$ (Figure (3)), the net force magnitude from pressure on the control volume at $\hat{s} = s_0$ will be

$$F_{ps} = p_{left}\delta A - p_{right}\delta A =$$

$$(p(s_0) - \frac{dp}{ds}\frac{\delta s}{2})\delta n \delta y - (p(s_o) + \frac{dp}{ds}\frac{\delta s}{2})\delta n \delta y = -\frac{dp}{ds}\delta s \delta n \delta y$$

Equation (9)

$$F_{ps} = -\frac{dp}{ds}\delta V.$$

The desired expressions for the gravitational force magnitude F_{gs} and pressure force magnitude F_{ps} on the control volume in the direction of the streamline are given by equations (7) and (9), respectively. Substituting these expressions in equation (4), we obtain

$$-\gamma \delta V \sin\theta \quad - \quad \frac{dp}{ds}\delta V \quad = \rho \delta V u \frac{du}{ds} \Rightarrow$$

Equation (10)

$$-\gamma \sin\theta \quad - \quad \frac{dp}{ds} \quad = \rho u \frac{du}{ds}.$$

We will express this relationship in a slightly different form by noting the following:

$u\frac{du}{ds} = \frac{1}{2}\frac{d(u^2)}{ds}$; and (see Figure 8)

$\sin\theta = \frac{dz}{ds}$.

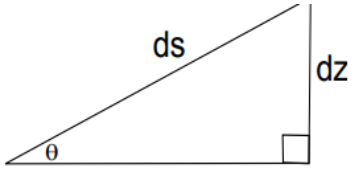


Figure 8. $\sin\theta = \dfrac{dz}{ds}$.

Thus, equation (10) becomes

$$-\gamma\frac{dz}{ds} - \frac{dp}{ds} = \frac{1}{2}\rho\frac{d(u^2)}{ds} \Rightarrow$$

Equation (11)

$$\frac{dp}{ds} + \frac{1}{2}\rho\frac{d(u^2)}{ds} + \gamma\frac{dz}{ds} = 0.$$

Finally, recalling that ρ is constant (incompressibility assumption A1), integrating both sides of equation (11) with respect to s gives

$$p + \frac{\rho u^2}{2} + \gamma z = \ constant\, along\, a\, streamline.$$

From equation (12) we see that there is a relationship along a streamline between the fluid pressure, particle speed and particle elevation. We have achieved our goal to obtain a basic form of *Bernoulli's equation*, a very powerful tool in fluid mechanics [Young et al. 2001].

When modeling gravity-fed water delivery systems, we will apply the following equivalent form of Bernoulli's equation obtained by dividing all terms in equation (12) by the specific weight $\gamma = \rho g$ (with $\frac{\rho}{\gamma} = \frac{1}{g}$):

$$\frac{p}{\gamma} + \frac{u^2}{2g} + z = Constant\, along\, a\, streamline.$$

The different terms in equation (13) represent three types of energy (pressure ($\frac{p}{\gamma}$), kinetic ($\frac{u^2}{2g}$) and potential (z)) expressed in units of length (**Exercise 2.3**). These terms are called *heads* as listed in **Table 3**. We will discuss application of these heads in the next section.

Table 3.
Types of heads.

Name of Head	Term in Bernoulli's Equation **(13)**
Pressure Head	$\frac{p}{\gamma}$
Velocity Head	$\frac{u^2}{2g}$
Elevation Head	z

12.6.3 Application to Gravity-Fed Water Delivery Systems

Basic Case

In the previous section, we showed how Bernoulli's equation describes a fluid particle moving along a one-dimensional streamline. Though it applies more generally than to water flowing through a closed pipe, Bernoulli's equation will now help us plan and model a simple, gravity-fed water delivery system. Note that the assumptions behind Bernoulli's equation are valid in this situation:

[A1] The fluid is incompressible- This assumption is good for liquids since they are less easily compressible than gases.

[A2] No sources or sinks of energy- All energy in this system comes from gravitational potential energy with no pumps, turbines, etc.

[A3] Steady flow- We will assume the spring keeps the source tank filled to capacity when the outlet faucet is open so that the water flow in the pipe is steady with time.

[A4] Inviscid flow- For our initial cases we will assume inviscid flow, meaning there is no friction, and energy is conserved.

We begin by considering the portion of a system in which water from a supply tank flows through a pipe to a single outlet faucet (Figure 9).

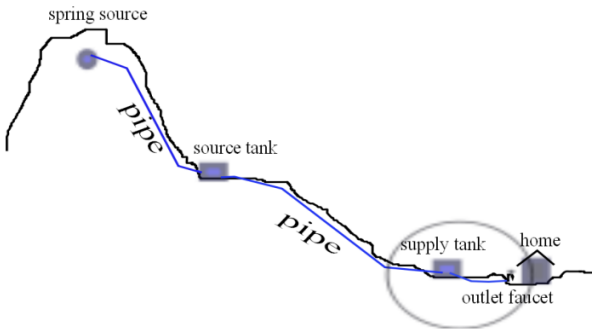


Figure 9. System segment from supply tank to outlet faucet.

When the outlet faucet is closed, water flow in the pipe is halted and hence water in the supply tank is conserved. On the other hand, Bernoulli's equation implies that pressure must build as we move down the pipe. Pressure in the case of a closed outlet is known as *static pressure*. To see why, let us consider the static case in which no new water enters the source tank and hence the water speed satisfies $u = 0$ throughout the system. In this case, Bernoulli's equation simplifies to

$$\frac{p}{\gamma} + z = C,$$

where C is constant. Referring to Figure 10, since at point A the pressure head is zero (i.e. equal to atmospheric pressure) and the elevation head is $z = h_1 + h_2$, the constant C must satisfy $C = h_1 + h_2$. At the base of the faucet (point B), the elevation head is equal to zero and hence the pressure head satisfies

Equation (14)

$$\frac{p}{\gamma} = h_1 + h_2 \;\Rightarrow\; p = (h_1 + h_2)\gamma.$$

If the static pressure build-up as we move along the streamline is too great, the delivery pipe could burst, as is considered

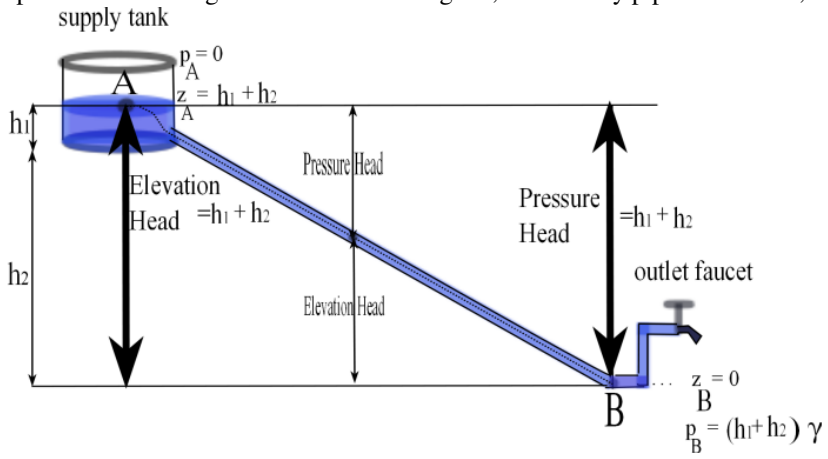


Figure 10. If the outlet faucet is closed, the elevation head at point A is converted into static pressure head at B. At each intermediate point along the streamline, the elevation and pressure heads have the same sum (at each point, the velocity head is zero, so this head is not included on the diagram).

in Exercise 3.1.2.

Break-Pressure Tanks

As mentioned in the previous section, if the flow of water is stopped in a gravity-fed system the static pressure inside the pipe must increase (increasing pressure head) as the pipe elevation decreases. If the pressure exceeds the pipe's pressure capacity, the pipe could burst (see Exercise 3.1 b). To prevent this type of system failure, a small *break-pressure tank* (see Figures (11) and (12)) is installed at some judicious point along the path of the pipeline. Such a tank, typically made of brick, relieves pressure by allowing the flow to discharge into open air space inside the tank.

Figure 11. A break-pressure tank installed by a team in Honduras.

In other words $p = 0$ (atmospheric pressure) at the water level inside the break-pressure tank. A simple shut-off float valve (similar to that used to stop water flow in a toilet) and safety overflow pipe keep the water from exceeding the tank's capacity so there is always open air space inside the tank. The break-pressure tank essentially becomes a new source tank for its outgoing pipeline Exercise 3.2.2.

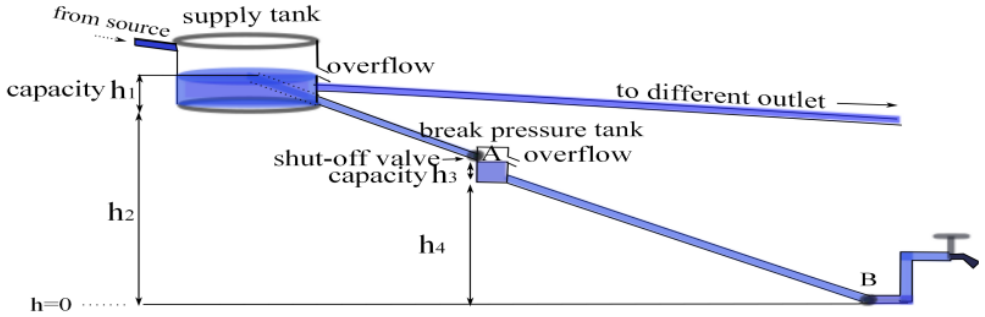


Figure 12. Simple system with break-pressure tank.

Water-Flow Rates

In the previous sections we discussed pressure build up in a static system in which the flow of water is stopped by an outlet faucet at the end of a pipe from the source tank. In this case the outlet faucet will be open and water will flow, so the pressure is referred to as *dynamic pressure* (as opposed to static). We make the distinction now since, in the case of viscous flow, the static and dynamic pressures will be different (see Section 4). In this section we will analyze the speed of the water as it flows from one tank to another (Figure 13). Similar reasoning can be used to determine the flow rate from a tank to a home outlet with an open delivery pipe.

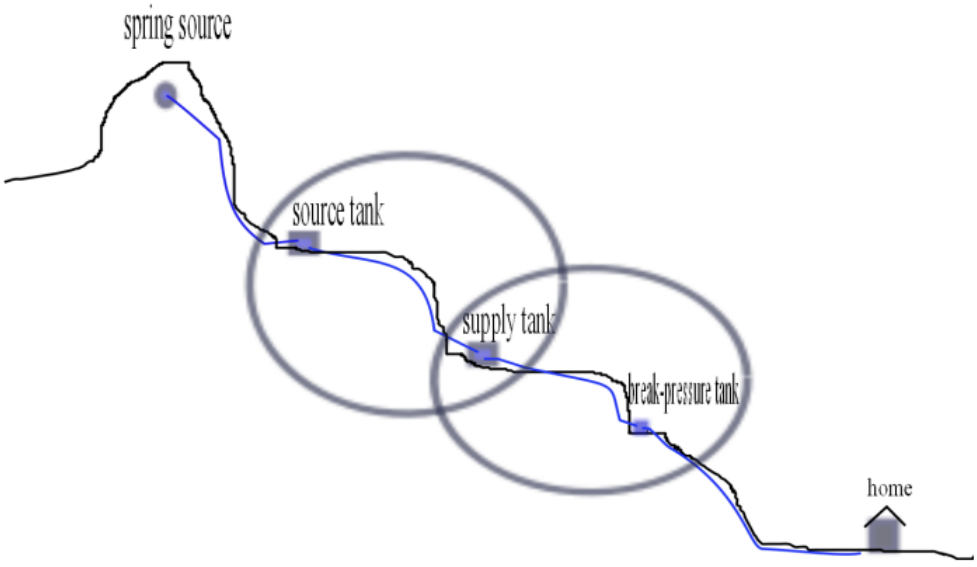


Figure 13. Tank-to-tank segments of the system.

Being able to project flow rates is important for a number of reasons [Tawney n.d.]. If the flow rate is too slow, sedimentation occurs inside the pipe causing it to clog. On the other hand, if the flow rate is too fast, the pipe wall may erode. For these reasons the average water velocity should remain between 0.7 and 3 m/s. Further, the *volumetric flow rate* or volume of water per unit time depends on the speed of the water. The ability to meet the demand for water is determined by the volumetric flow rate.

Consider the gravity-fed water delivery system shown in **Figure 14**, in which a source fed pipe with radius r_1 supplies a tank at a rate u_0 so that the tank remains at height h_1 above the delivery pipe; and the delivery pipe with radius r_2 and vertical drop h_2 meters connects the first tank to a second tank with capacity h_3.

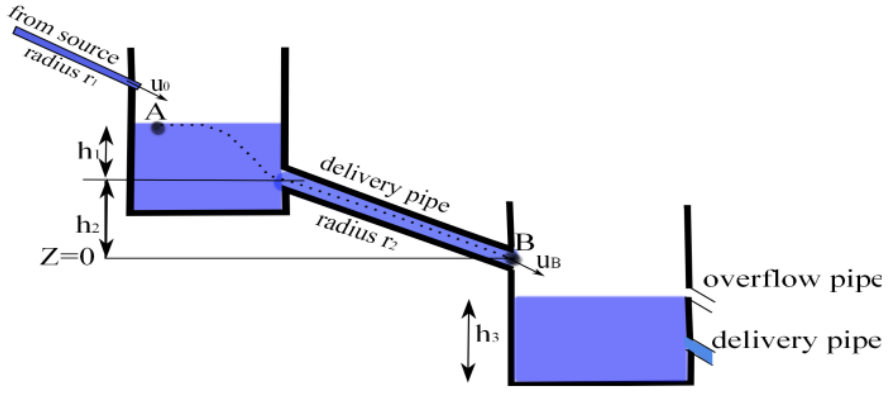


Figure 14. Gravity-fed systems convert elevation head (point A) to velocity head (point B).

We find the rate u_B at which water exits the delivery pipe into the receiving tank as follows:

at point A, $u = 0, p = 0, z = h_1 + h_2$; and

at point B, $u = u_B, p = 0, z = 0$.

Applying Bernoulli's equation, we obtain

$$\frac{p_A}{\gamma} + \frac{u_A^2}{2g} + z_A = \frac{p_B}{\gamma} + \frac{u_B^2}{2g} + z_B \Rightarrow$$

$$0 + 0 + (h_1 + h_2) = 0 + \frac{u_B^2}{2g} + 0 \Rightarrow u_B = \sqrt{2g(h_1 + h_2)}.$$

12.6.4 Viscous Flow in Pipes

Head Loss

To this point we have assumed inviscid or frictionless flow, but now we will consider how friction affects a gravity-fed system. In the inviscid case all energy was conserved, but now some of the initial gravitational energy in the pipe will be lost to friction as it flows through the streamline. This loss of energy is known as *head loss*, which is measured in meters and denoted as h_f. Since all energy will either be conserved or lost to friction, Bernoulli's equation becomes

Equation (15)

$$\frac{p}{\gamma} + \frac{u^2}{2g} + z + h_f = Constant\, along\, a\, streamline.$$

It is important to note that in the case of a closed outlet faucet the velocity and, consequently, the head loss, will both be zero. This is the case of static pressure and is the same for inviscid and viscous flow. However, when the outlet is open energy will be lost along the streamline. Since the velocity and elevation heads will be unchanged by friction all head loss will come from the dynamic pressure head in the form of a pressure drop Δp. In other words,

$$Dynamic\, Pressure = StaticPressure - h_f.$$

Note that if a pipe can withstand a system's static pressure case it will be able to withstand the dynamic pressure (**Exercise 4.1**).

Reynold's Number

Before we continue, we need to look closer at the friction occurring between the water and the pipe wall. In particular, we will investigate how the fluid velocity vectors on a cross-section of the pipe are affected by friction.

The effect of friction is related to the type of flow, classified into one of three types (see Figure 15):

laminar– streamlines are smooth;

turbulent– streamlines exhibit random fluctuations with pronounced nonlinear characteristics; and

transitional– streamlines change between laminar and turbulent characteristics.

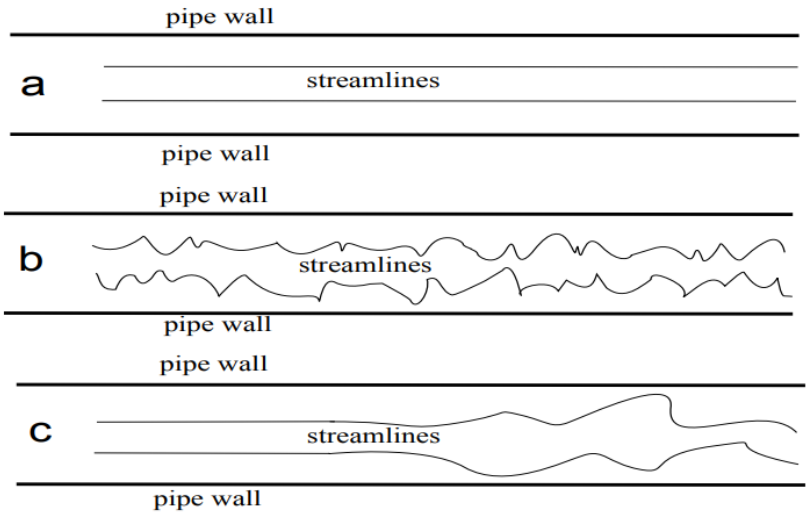
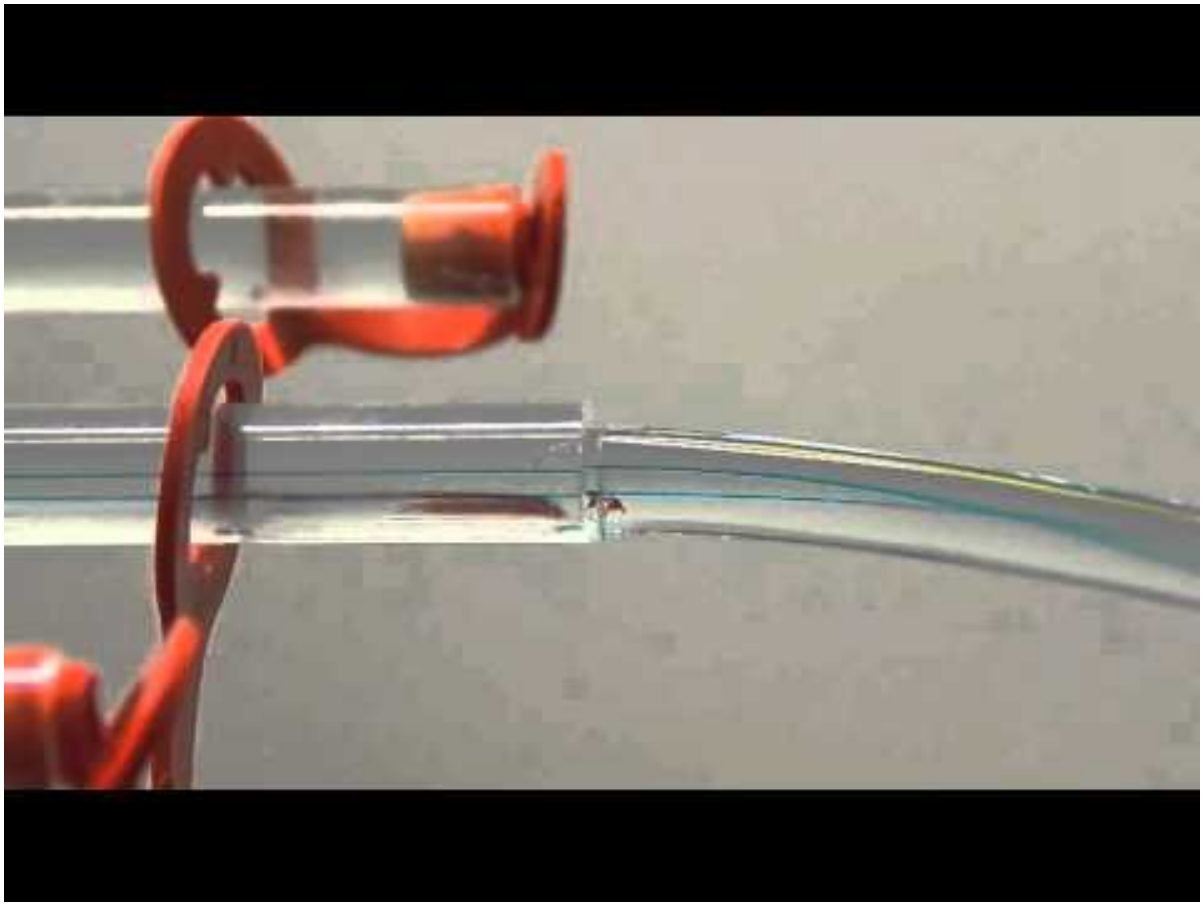


Figure 15. Qualitative examples of streamlines for a) laminar; b) turbulent flow; c) transitional flow.

Note

Please visit the web version of this section to engage with this interactive resource.

```
# This video shows the transition between laminar and turbulent flow.
# "Why Laminar Flow is AWESOME" YouTube, uploaded by SmarterEveryDay, 27 January 2019,
 ↪ https://youtu.be/y7Hyc3MRKno?si=0jG3mpGSVrHlN_4p. Permissions: YouTube Terms of␣
 ↪Service

from IPython.display import YouTubeVideo
YouTubeVideo('rn9y1CSoFZs')
```



The standard way to distinguish between the different kinds of flow is using the *Reynolds number*. Denoted Re, the Reynolds number is defined as

$$Re = \frac{\rho \bar{u} D}{\mu},$$

where ρ is the fluid density (mass/volume), $\bar{u}$ is the average speed on a cross section, D is the pipe diameter, and μ is the *dynamic viscosity*, a parameter accounting for the fluid's resistance to flow ($\mu_{H_2O} = 0.001 \frac{kg}{m \cdot s}$ at $20^o C$).

For a round pipe, the type of flow relates to the Reynolds number approximately as follows:

$Re < 2100 \rightarrow$ Laminar flow: flow follows a smooth path;

$2100 \leq Re \leq 4000 \rightarrow$ Transitional flow: flow is a mix of laminar and turbulent characteristics; and

$4000 < Re \rightarrow$ Turbulent flow: flow is irregular with random changes in direction.

Laminar Flow

Laminar flow is the easiest to analyze, and allows us to find a mathematical expression for the effects of friction on the flow. It can also be necessary for studying transitional flow since it is a combination of laminar and turbulent characteristics. However, it is worth noting that laminar flow is very rare in gravity-fed water delivery systems due to the velocities and diameters used (**Exercise 4.2.4**). For our derivation of velocity for a laminar flow we will begin by looking at a homogeneous, circular pipe with constant radius. To this end, we introduce cylindrical coordinates as follows (see Figure 16):

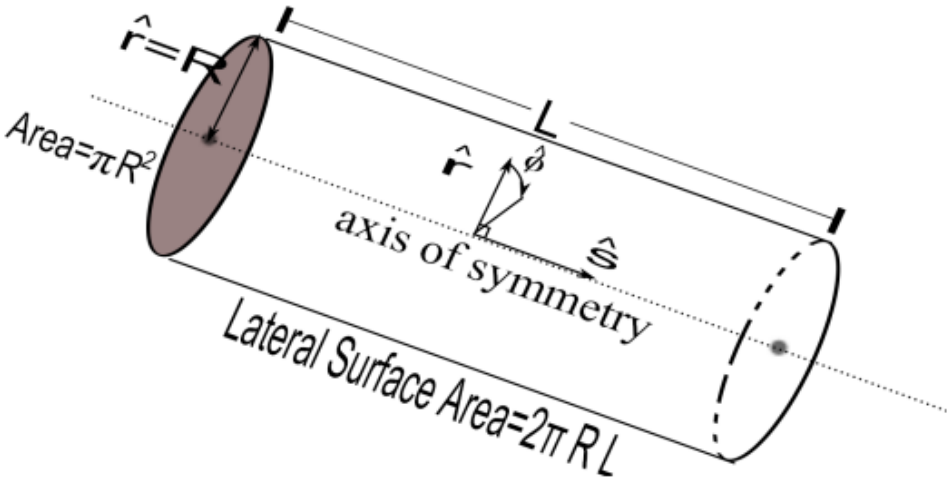


Figure 16. Cylindrical coordinates used in pipe-flow analysis.

$\hat{s}$= position on a streamline along the central axis of symmetry;

$\hat{r}$= radial distance from the central axis of symmetry;

$\hat{\phi}$= not needed since we assume radial symmetry (as in the 2D pipe representation in Figure 17.)

We assume for our steady, incompressible viscous flow that velocity vectors are always perpendicular to each normal cross section of the pipe. That is, velocity is always parallel to the central axis of symmetry. It is important to note that velocity is no longer constant on a normal cross-section. Friction occurs where the water contacts the pipe, creating a cross-sectional *velocity profile*, as shown in **Figure 17**. Note, in particular, that the velocity of the water is $\vec{0}$ at the pipe wall (*no-slip condition*) and is a maximum along the central axis of symmetry of the pipe ($\hat{r} = 0$).

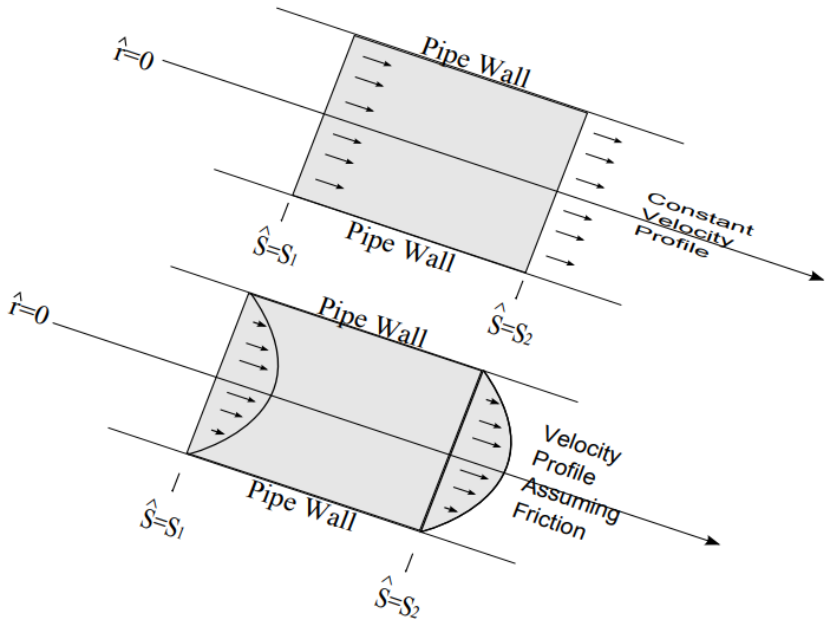


Figure 17. The constant cross-sectional velocity profile for inviscid flow (top figure) changes due to friction (bottom figure).

Note that the velocity profile is assumed to be the same on every cross-section. In other words, there is no fluid acceleration.

A velocity profile is determined by its *velocity gradient*, meaning the rate of change $\frac{du}{dr}$ of the magnitude of the velocity $u = \mid \vec{u} \mid$ with respect to $\hat{r}$. Friction is described by the *shear stress* $\tau(r)$, which gives the magnitude of the frictional force per unit (lateral surface) area r units from the central axis of symmetry. For a *Newtonian fluid* (such as water), stress is related to the velocity gradient by the equation

$$\tau(r) = -\mu \frac{du}{dr},$$

where μ is a constant called the *dynamic viscosity*. The negative sign indicates that the frictional force opposes the direction of fluid motion.

To derive the velocity profile we consider a concentric, cylindrical control volume (i.e. mathematical construct) with radius r, cross-sectional area $A_{cs} = \pi r^2$, and lateral surface area $A_L = 2\pi r L$ (**Figure 18**).

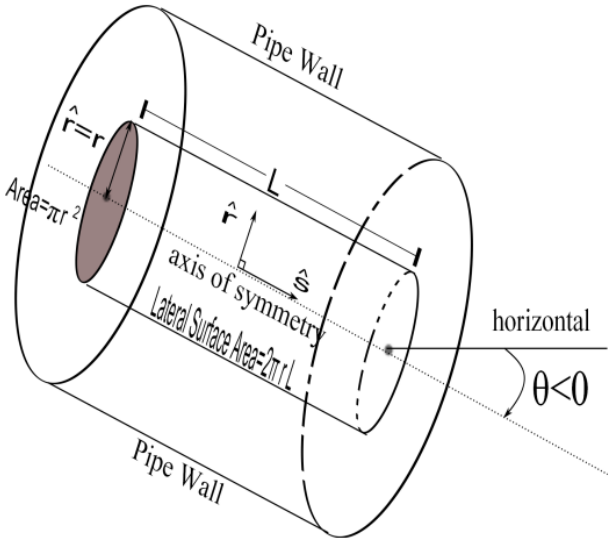


Figure 18. The inner cylinder is the control volume used to obtain the velocity profile $u(r)$.

By Newton's 2nd law, the total force (gravitational, pressure, and friction) acting in the $\hat{s}$ direction on this volume must be zero since there is no acceleration in that direction:

Equation (18)

$$F_{gs} + F_{ps} + F_{friction} = ma_s = 0.$$

The component of the gravitational force F_{gs} on the control volume in the $\hat{s}$ direction is given by $F_{gs} = -\gamma \pi r^2 L \sin\theta$ (Exercise [4.3.1]). The component of the pressure force is given by $F_{ps} = -\Delta\overline{p} A_{cs} = -\Delta\overline{p}\, \pi r^2$ where $\Delta\overline{p} = \overline{p}_{right} - \overline{p}_{left}$ is the difference between the average pressures on the right and left faces of the control volume. The frictional force is given by $F_{friction} = -\tau(r) A_L = -\tau(r) 2\pi r L$. It follows from equation (18) that

$$-\gamma \pi r^2 L \sin\theta - \Delta\overline{p}\, \pi r^2 - \tau(r) 2\pi r L = 0 \Rightarrow$$

$$\tau(r) = -(\frac{\gamma L \sin\theta + \Delta\overline{p}}{2L})r.$$

Recalling our assumption that stress is simply proportional to the velocity gradient ($\tau(r) = -\mu \frac{du}{dr}$), we have

$$\frac{du}{dr} = (\frac{\gamma L \sin\theta + \Delta\overline{p}}{2\mu L})r \Rightarrow$$

$$u(r) = (\frac{\gamma L \sin\theta + \Delta\overline{p}}{4\mu L})r^2 + C_1,$$

for some constant C_1. Noting that the velocity at the pipe wall is zero ($u(R) = 0$) gives $C_1 = -(\frac{\gamma L \sin\theta + \Delta\overline{p}}{4\mu L})R^2$. We conclude that the velocity profile for laminar viscous flow in a pipe must be parabolic:

Equation (20)

$$u(r) = -(\frac{\gamma L \sin\theta + \Delta\overline{p}}{4\mu L})R^2[1 - (\frac{r}{R})^2] = u_0[1 - (\frac{r}{R})^2$$

where $u_0 = u(0) = -(\frac{\gamma L \sin\theta + \Delta\overline{p}}{4\mu L})R^2$ is the speed along the central axis of symmetry.

In Exercise [4.3.2] you are asked to show that the corresponding volumetric flow rate is $Q = \frac{\pi R^2 u_0}{2}$. Note that the maximum flow rate u_0 is twice the constant speed of inviscid flow [Young et. al. 2001], so that the volumetric flow rate Q remains the same for laminar and inviscid flow (as one might predict from the continuity equation for incompressible flow).

Turbulent Flow

Modeling of turbulent flow is more complex than laminar flow in many respects [Young et. al. 2001], including:

the velocity gradient for turbulent flow changes with the position $\hat{s}$ of the cross section. (In this Module, for both inviscid and laminar flow, we only consider the case where the velocity gradient is the same on each cross section (i.e. fully developed flow));

the shear stress τ for turbulent flow is no longer proportional to the velocity gradient and depends on both the dynamic viscosity μ and the mass density ρ of the fluid. (For laminar flow, shear stress is independent of fluid density); and

the pressure drop Δp for turbulent flow depends on a roughness parameter ϵ of the pipe wall. (For laminar flow, pressure drop is independent of pipe roughness).

The above considerations imply that for steady, incompressible turbulent flow, the pressure drop Δp in a round pipe with constant diameter may be described by a function of the form

Equation (21)

$$\Delta p = F(\bar{u}, R, L, \epsilon, \mu, \rho)$$

where $\bar{u}$ is the average velocity on a cross section, R is the pipe radius, L is the length of the pipe, ϵ is a roughness parameter of the pipe wall, μ is the dynamic viscosity, and ρ is the density. (Other factors affecting pressure, such as water temperature, suspended particles, or dissolved gases are considered to be negligible, so equation (21) serves as a good approximation)

For steady, incompressible flow through a pipe of length L and diameter $D = 2R$, the head loss $h_f(L)$ (i.e. loss in total energy due to friction) is dependent on a *friction factor* f according to the *Darcy-Weisbach* formula [Young et. al. 2001]

Equation (22)

$$h_f(L) = f\frac{L}{D}\frac{\bar{u}^2}{2g}.$$

Here the average speed $\bar{u}$ is the same on each cross-section by the continuity equation **(Exercise 3.3)**. The friction factor f is assumed to be a function of the form

$$f = G(Re, \frac{\epsilon}{D})$$

where Re is the Reynolds number and $\frac{\epsilon}{D}$ is called the relative roughness. The friction factor f is related to the Reynolds number Re and relative roughness ϵ/D, via the equation

$$\frac{1}{\sqrt{f}} = -2.0\log_{10}(\frac{\epsilon/D}{3.7} + \frac{2.51}{Re\sqrt{f}}),$$

whose solutions are compiled in tables or displayed graphically as *Moody charts* [Young et. al. 2001]. The latter has the qualitative form shown in Figure (20).

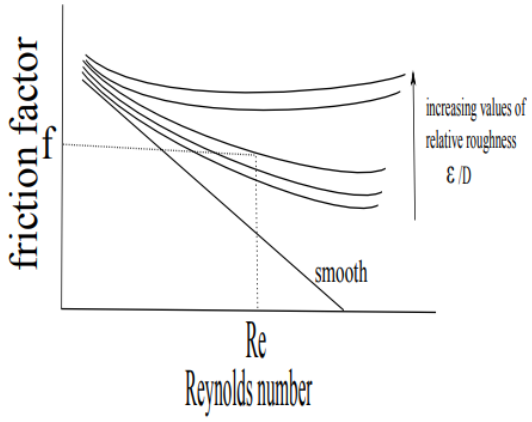


Figure 20. Qualitative Moody chart displaying the friction factor f as a function of Reynolds number (Re) and relative roughness ($\frac{\epsilon}{D}$). Charts of this type constructed from experimental data are used in practice to determine f for turbulent flows.

Pipe Selection

We have introduced a number of important parameters related to pipe selection, including diameter, average cross-sectional speed, volumetric flow-rate, head loss due to friction, pressure capacity, Reynolds number and the friction factor, as summarized in Table 4. In this section, we discuss how relatively small changes in pipe diameter can lead to dramatic changes in the system design.

Table 4.

Data for **Exercise 3.2.1**.

horizontal distance (m)	elevation (m)
150	977
300	955
450	940
600	886
700	895

Figure (21) qualitatively shows a segment of a water delivery system for which three different pipes are being considered. The static head line shows the initial total energy available due to gravitational potential energy. The *hydraulic grade line (HGL)* for each pipe shows the amount of energy remaining after head loss as the water flows along the streamline.

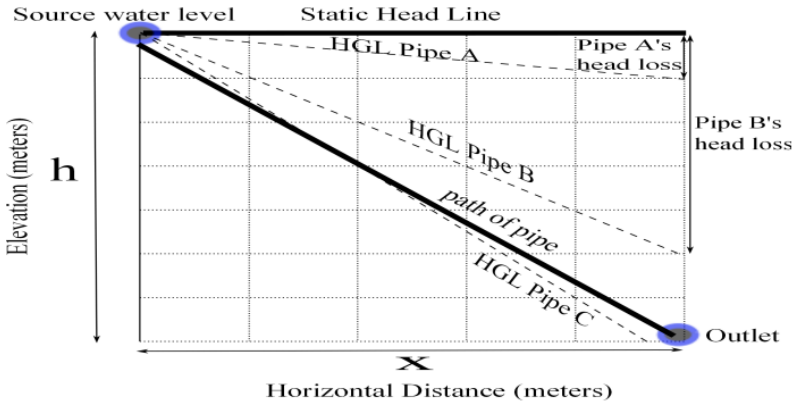


Figure 21. Hydraulic grade line (HGL) diagram for various pipe specifications.

Table 5 (based on Appendix C of [Tawney n.d.]) shows the meters of head loss due to friction (per 100 meters of pipe) for various diameter pipes and flow rates. Note that flow rates that are too fast or too slow for practical use are excluded ($> 3m/s$ causes too much erosion inside the pipe, while $< 0.7m/s$ sediment from water settles and can block the pipe).

Table 5.
Some important parameters for a pipe with length L.

Formula	Unit	Module Sections
Pipe diameter D	m	3.3
Average flow rate $\bar{u}$	m/s	3.3, 4.2, 4.3
Volumetric flow rate Q $Q = \pi \left(\frac{D}{2}\right)^2 \bar{u}$	m^3/s	3.3, 4.2
Head loss h_f $h_f = f\frac{L}{D}\frac{\bar{u}^2}{2g}$	m (m)	4.3
Required minimum pressure capacity C $C = 0.1\gamma h$	bar = (N/cm^2)	3.1, 3.2
Reynolds number $Re = \frac{\rho \bar{u} D}{\mu}$	dimensionless	4.1, 4.3

Observing that head loss and pressure capacity both vary inversely with pipe diameter, the three pipes in Figure 21 may be categorized as follows:

Pipe A: Head loss is the smallest, suggesting that this pipe has the largest diameter D and volumetric flow-rate Q. Pipe A can thus service a greater demand for water, and friction effects are only a small factor in impeding water flow from reaching the destination outlet. However, a larger size pipe has a lower pressure capacity C, so this pipe is most at risk of failure if the flow is halted by a closed outlet. In other words, a break-pressure tank is most likely needed for this pipe to prevent the static pressure from getting too high. (A break-pressure tank is required if the pipe's allowable pressure head is less than the initial elevation head (h) minus the head loss (h_f));

Pipe B: This pipe has an intermediate pipe diameter and flow-rate, and as such, can meet an intermediate demand and also has an intermediate pressure capacity. A break-pressure tank may or may not be needed. Even with significantly greater head loss than pipe A, there is still enough available energy to transport the water all the way to the outlet; and

Pipe C: This pipe, with the smallest pipe diameter and volumetric flow-rate, would have the highest pressure capacity and thus is most likely to eliminate the need for a break-pressure tank. However, it can only meet a limited demand. Furthermore, since the pipe has a smaller diameter, the water will move through it with a higher velocity and lose more energy to friction. More energy goes into the head loss, making the HGL more steep and causing the HGL to fall below

the pipe level. Where the HGL falls below the pipe level, there is insufficient energy available to move the water in the pipe to the outlet. In this case, a pump would have to be employed. (For systems empowered by gravity alone, the HGL should never fall below the pipe level since this results in negative pressure (to satisfy Bernoulli's equation) which will prevent motion and can suck air or dirt into the pipe. Ideally the pipe will be built at least 5 meters below the HGL.)

Varying diameters in water-delivery systems between roughly 25 and 50 mm in diameter can result in these major differences in system performance ([Tawney n.d.]), thus highlighting the importance of careful system design before beginning construction.

12.6.5 Designing a System

In this section, we will apply the ideas we have presented to the problem of designing a gravity-fed system for the village of Nan Mand in Micronesia. Using data from [Mogenson, 1996], we note that:

the population of 400 grows at an average rate of 2% annually;

the water demand is estimated at 416 liters per person per day; and

the system will flow from a source tank at an elevation of 195 m to an outlet faucet 2220 m away at 5 m, and the elevation survey between these points returns:

Table 7.
Survey of elevations from source tank to outlet faucet in Nan Mand, Micronesia.

horizontal distance (m)	elevation (m)
213	177
701	134
1128	116
1524	91
1890	46

Our system design specifications include:

the system will meet the village's need for the next 20 years;

the volumetric flow rate will be maintained with an acceptable water velocity;

the location and number of break pressure tanks (if any) will be specified;

head loss will be analyzed via an HGL diagram to ensure that the pipe stays below the HGL (hydraulic grade line); and

the pipe diameter will be the same for the whole system for the sake of simplicity.

At its current rate, the village will have a population of $400 * 1.02^{20} = 594$ in 20 years. With a water demand of 416 liters/person/day the system must supply 2.86 liters/sec, which we convert to 0.00286 m^3/s (1 m^3 =1000 liters). Since we have a fairly large demand and a small change in elevation (i.e. less pressure buildup) we will first try a 50 mm diameter pipe. We will use the continuity equation

$$Q = \bar{u} * A \rightarrow \bar{u} = \frac{0.00286 m^3/s}{\pi * 0.025^2} = 1.457 m/s,$$

which is within the velocity bounds of 0.7 and 3 m/s (otherwise a new pipe diameter would need to be selected). Next we consider the static pressure limitations. The greatest pressure will be at the outlet faucet at an elevation of 5 m when the outlet is closed. The difference in elevation from the outlet to the source tank (190 m) gives us a pressure head of 190 m. We convert this to pressure as follows:

$$Pressure\, Head = \frac{p}{\gamma} \rightarrow p = 190 * 9789 = 1860 \frac{kN}{m^2}.$$

This static pressure exceeds the 50 mm pipe's maximum operating pressure of 1144.5 $\frac{kN}{m^2}$, so at least one break pressure tank will be needed. If one is placed at an elevation of 91 m (horizontal distance 1128 m), the first and second maximum static pressures will be 1019.2 $\frac{kN}{m^2}$ and 842.8 $\frac{kN}{m^2}$, respectively. Both of these are within the pipe's rating, so no more break pressure tanks will be needed. The head loss per 100 m is given in **Table 5**, and since our flow rate of 2.86 l/s isn't included we will use the closest value of Q (3 l/s) for this step. Also, since the pipe is at a small angle, we will estimate the pipe length as the horizontal distance. With this in mind our rate of head loss is 6.04 m per 100 m of horizontal distance (x). Now we will look at the dynamic pressure (outlet open) to make an HGL diagram using equation (13). The different heads are as follows:

Velocity Head ($\frac{\bar{u}^2}{2g}$)-constant, $= \frac{1.457^2}{2*9.8}$ =0.108 m

Elevation Head (z)-elevation of the current point in meters

Head Loss (h_f)-increases at constant rate with horizontal distance, $= \frac{x}{100} * 6.04$

Pressure Head ($\frac{p}{\gamma}$)-dynamic pressure while pipe is flowing, $= z - \frac{\bar{u}^2}{2g} - h_f$

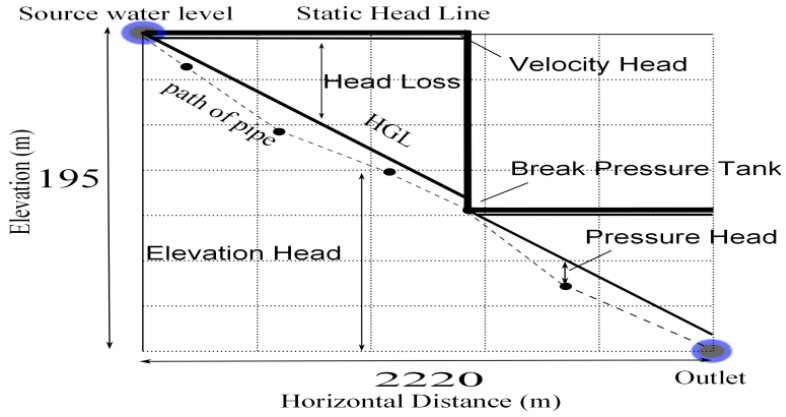


Figure 22. HGL diagram for Nan Mand village.

In Figure 22, the slope of the Hydraulic Grade Line is the rate of head loss, so the HGL has an equation of $HGL = 195 - 0.108 - 0.0604x$. Since the pipeline remains below the HGL at all points, we can be satisfied that this design meets all conditions. If the pipe went above the HGL, we would have to re-design the pipe with a larger diameter to reduce head loss and prevent negative pressures.

12.6.6 Case Study: Honduras

A service learning project is an invaluable enrichment or motivation to the modeling of a gravity-fed water delivery system as described in this Module. For example, the nation of Honduras is one of the poorest in the Western Hemisphere, and its rural areas have historically had little access to clean water. In 2002, the World Health Organization (WHO) estimated 2,500 people die in Honduras each year due to lack of clean water. Gravity-fed water supply systems are cheap and low-maintenance, and the availability of mountain springs in Honduras makes such systems ideal to meet real needs. Indeed, water supply systems that use gravity correspond to 93% of all constructed systems in Honduras.

The *Honduras Project* (HP) is a student-led organization at Wheaton College which, since 1982, has helped bring potable water to under-served villages by installing gravity-fed systems. HP partnered for many years with the late Arnoldo Alvaraez, a Honduran civil engineer with years of experience designing, installing and maintaining systems. Throughout the year, the students raise funds through community projects and support letters. During spring break, the group travels to a Honduran village to help dig trenches, build tanks, and lay pipes.

Figure 23. Each year, Honduras Project assists a rural Honduran village to install a gravity-fed water delivery system.

HP, and programs like it, have helped Honduras' water coverage improve significantly over the last few decades (Figure 24). These kinds of programs can be replicated by other schools or organizations. If you would like to learn more about the Honduras Project, you can visit their website at https://www.wheaton.edu/life-at-wheaton/ministry-and-evangelism/students/ministry-teams/honduras-project/.

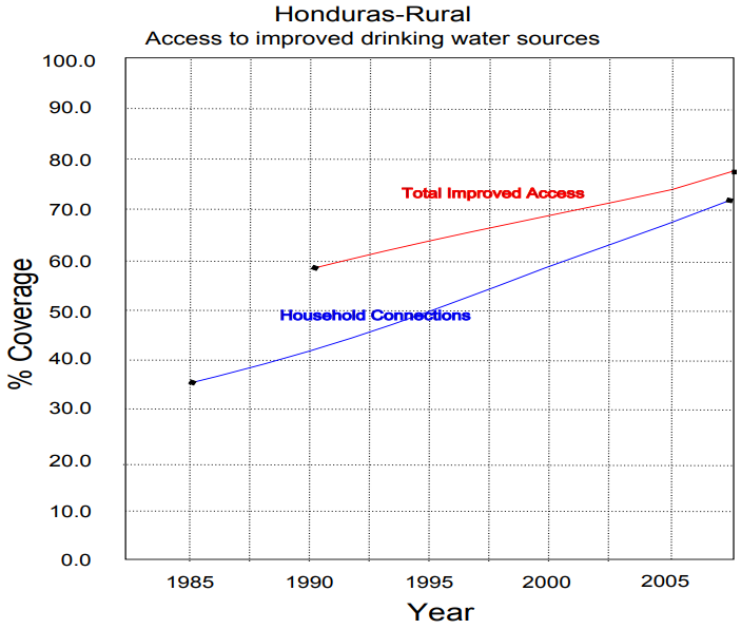


Figure 24. Improved access to water in rural Honduras [WHO/Unicef 2010].

12.6.7 Further Directions

In this Module, we have derived Bernoulli's equation and then applied it to the modeling of pressure build-up and flow rates in a basic gravity-fed water delivery system. We then introduced break-pressure tanks, computed the velocity profile for laminar viscous flow, and described qualitatively friction effects for turbulent flow.

There are both practical and mathematical considerations leading to models of increasing complexity which have not been discussed in this chapter, including, but not limited to

- study of the effects on pressure and velocity if the main pipe splits into the smaller pipes of a distribution system;
- consideration of daily peak demand for water;
- evaluation of the water source, determining its quality and supply capacity;
- computation of vertical drop using surveying instruments;
- determination of demand in the village being serviced, including peak periods and population growth;
- inclusion of air-release valves within pipe systems;
- selection of attachments (eg. faucets) in delivery to individual homes;
- discussion of system maintenance and safety factors such as long-term use of PVC pipes;
- consideration of pressure variation within curved pipes;
- inclusion of other factors contributing to head loss such as suspended particles, dissolved gases, and water temperature;
- incorporation of non-steady and transitional flow analysis including head-loss through valves etc; and
- application of chaos theory, fractal geometry, and numerical analysis to turbulent flow.

Those interested in further study may consult Tawney [n.d.] and ACF [2008] for practical considerations, Nakayama and Boucher [1999] or Young et. al. [2001] for general background in fluid mechanics, Chin [2000] for general background in water-related engineering problems, and Jordan [1980] for a more in-depth discussion of gravity-fed water delivery systems.

Practical resources include:

Graphing calculators can easily be used to compare elevations with the HGL by entering the elevations into the stat plot and then graphing the HGL equation;

https://www.engineeringtoolbox.com/pvc-cpvc-pipes-pressures-d_796.html gives the maximum operating pressures of PVC pipes;

https://www.efunda.com/formulae/fluids/calc_reynolds.cfm provides a useful Reynolds number calculator with several different kinds of units.

12.6.8 Exercises

Exercises

Section 2

2.1) Express in terms of γ, δV, and $\frac{dz}{ds}$ the component of the gravitational force F_{gn} in the $\hat{n}$ direction.

2.2) Recalling that the derivative is obtainable as a limit of a difference quotient, obtain the control volume pressure estimate $p_{left} = p(s_0 - \frac{\delta s}{2}) \approx p(s_0) - \frac{dp}{ds}\frac{\delta s}{2}$.

2.3) Let L be the dimension for length, M for mass, and T for time. Check the consistency of the dimensions in Equation (13) (terms which are added must have the same dimension).

Section 3

3.1.1 a) Referring to Figure (10), given that $h_1 = 2$ m, $h_2 = 9$ m, and water has a specific weight $\gamma = 9.789\frac{kN}{m^3}$, compute the pressure at point B in $\frac{kN}{m^2}$, assuming the tap is closed (i.e. static pressure).

b) We are designing the same system as in part a using US units. We are given $h_1 = 6$ ft., $h_2 = 30$ ft., and water's specific weight $\gamma = 62.3\frac{lb}{ft^3}$. Compute the pressure at point B in psi (pounds per square inch), assuming the tap is closed.

3.1.2) Given that a 2.5 cm diameter PVC pipe has a maximum operating pressure of $1862\frac{kN}{m^2}$, how many meters below the base of the source reservoir tank could the base of the faucet be safely located assuming the water level in the source reservoir never exceeds 3 meters? (All maximum operating pressures of PVC pipes are for schedule 40 pipes [Engineeringtoolbox n.d.])

Table 4.
Data for **Exercise 3.2.1.**

horizontal distance (m)	elevation (m)
150	977
300	955
450	940
600	886
700	895

a) At what horizontal distance will the pressure be the greatest when the faucet is closed?

b) If a 10 cm diameter PVC pipe (with a maximum operating pressure of 917 $\frac{kN}{m^2}$) was used, how many intermediate tanks would be needed to break pressure? (Water has a specific weight $\gamma = 9.789\frac{kN}{m^3}$.)

[3.2.1] A water delivery system is being designed based on the land surveyed in Tawney [2000] with a source tank at an elevation of 1000 m and an outlet faucet located at a horizontal distance of 850 m away at an elevation of 906 m. A survey for the proposed pipe path resulted in the following data:

3.2.2 Referring to Figure (12), assuming that the supply and break pressure tanks are filled to capacity, compute the water pressure in the pipe at point A (closed shut-off valve) and also at point B (base of the closed faucet). By how much is the pressure at the base of the faucet be reduced with the inclusion of the break pressure tank?

[3.3.1] The village of Mundenda, Zimbabwe is made up of 70 households with an average of five people per household whose minimum requirement for healthy living is 30 liters per person per day [Manyanhaire, 2009]. Find the volumetric flow rate Q (in liters/s) needed to keep its supply tank filled to capacity during an average demand period.

[3.3.2] If the pipe radius narrows (or alternately, widens), the rate of incompressible, steady fluid flow will increase (resp. decrease). To see why, consider two points $\hat{s} = s_1, \hat{s} = s_2$ on a streamline. Let A_1, A_2 be the respective normal cross-sectional pipe areas (in m^2) and u_1, u_2 be the respective (average) speeds of the water flow through A_1, A_2. The *continuity equation* (see Chin [2000]) states that $u_1A_1 = u_2A_2 = Q$, where the constant Q is the volumetric flow rate.

a) Suppose water flowing through a 5 cm (2 inch) diameter pipe has a velocity head of 3 m (10 ft). According to the continuity equation, what would the velocity head be if the pipe narrows to a 4 cm (1.5 inch) diameter? (Assume that $g = 9.8\frac{m}{s^2}$).

b) For the system shown in Figure 14, find the value for the spring supply rate u_0 (in terms of h_1, h_2, r_1 and r_2) in order to maintain steady flow at B.

Section 4

[4.1] A water source at an elevation of 50 m is flowing to an outlet faucet at 5 m. In the middle of the pipe at 30 m elevation you measure the velocity at 2.5 m/s. If the head loss to that point is 25 m, what does this mean for the pressure at that point? Recall that atmospheric pressure is our reference with p=0.

[4.2] Show that the Reynolds number is dimensionless.

[4.3.1] Show that $F_{gs} = -\gamma \pi r^2 L \sin\theta$.

[4.3.2] Show that for a circular pipe with radius R, the laminar, viscous flow rate (volume per unit time) through a cross section is $Q = \frac{\pi R^2 u_0}{2}$ (Hint: Using polar coordinates, integrate the velocity profile $u(r)$ given in equation (20) over a disc with radius R).

[4.3.3]

a) How does Figure 19 (adopted from [Nakayama, 1999]) account for the changing pipe diameter? Head loss due to friction?

b) How is Bernoulli's equation for inviscid flow modified to include head loss?

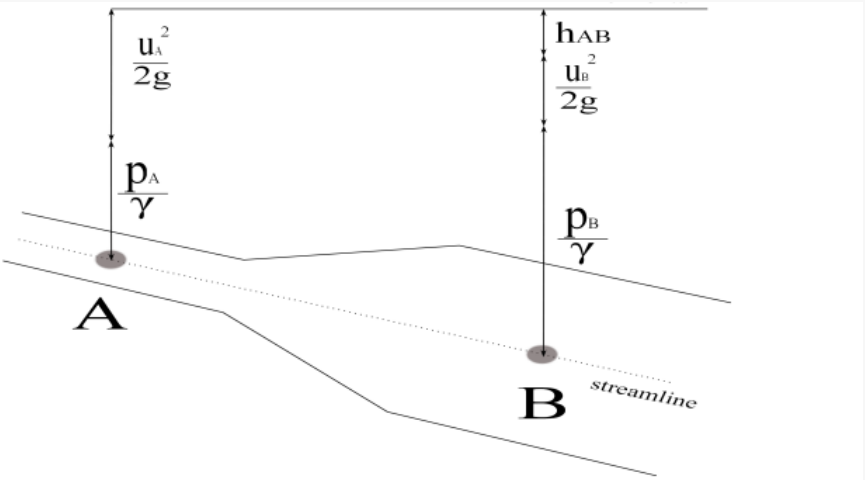


Figure 19. Figure for **Exercise 4.2.3**. (Adapted from Nakayama and Boucher [1999].)

[4.3.4] A pipe is being constructed to transport water at an average speed of 2 m/s. What is the largest possible diameter of pipe that would still have laminar flow? What would be the volumetric flow rate from this pipe? Is this realistic? Recall that $\rho_{H_20} = 998.2 \frac{kg}{m^3}$.

[4.4.1] Given that PVC is smoother than an iron pipe (i.e., PVC has a smaller relative roughness $\frac{\epsilon}{D}$ than iron pipe), which of the two pipes has the greater friction factor f for a given Reynolds number Re?

[4.4.2] Assuming the maximum speed for turbulent flow is between that of inviscid and laminar flows, make a qualitative sketch showing the (symmetric) velocity profiles on a cross-section for these three types of flows.

[4.5.1] For a 25 mm diameter PVC pipe, a volumetric flow rate Q between .35 and 1.4 L/s is recommended by **Table 5**. Find the Reynolds number Re and head loss associated with the minimum and maximum flow rate.

[4.5.2] Suppose you are considering Pipe X for the pipeline shown in **Figure 21**. You are given Pipe X's diameter D, length $L = \sqrt{h^2 + x^2}$, friction factor f, and relative roughness $\frac{\epsilon}{D}$. Explain how you would obtain the Reynolds number Re, average flow rate $\bar{u}$, volumetric flow rate Q, head loss h_f, and hydraulic grade line (HGL).

Section 5

[5.1] You are designing a gravity-fed water delivery system for the village of Tesuque Pueblo, New Mexico. The village has 980 people with an annual growth rate of 1.6 % and an expected use of 322 liters per person per day. The Barranca Road source tank is at an elevation of 2255 meters and the outlet faucet is 15.3 km away at an elevation of 1920 m (Leder, 2001). An elevation survey gives us the following data:

Table 8.
Survey of elevations from source tank to outlet faucet in Tesuque Pueblo, New Mexico.

horizontal distance (m)	elevation (m)
2415	2170
4225	2130
5625	2110
10050	1980
13265	1950

Design a system that can meet the average demand for the next 10 years with no upgrades. Choose a pipe diameter listed in Table 6 that meets the flow rate and velocity restrictions and assume all pipe to be the same diameter

References

ACF International. 2008. *Design, Sizing, Construction and Maintenance of Gravity-Fed Systems in Rural Areas, Module 2.* France: ACF.

Bentley, Jessica, et al. 2002. *Sustainable Water Development for the Village of Miramar, Honduras.* Water Resources Program, The University of New Mexico. http://repository.unm.edu/bitstream/handle/1928/10483/wrp-5.pdf?sequence=1

Chin, David. 2000. *Water-Resources Engineering*. Upper Saddle River, NJ: Prentice Hall.

eFunda.com. 2011. Reynolds Number Calculator http://www.efunda.com/formulae/fluids/calc_reynolds.cfm

EngineeringToolbox.com. 2011. http://www.engineeringtoolbox.com

Hofkes, E.H. 1981. *Small Community Water Supplies*. New York: John Wiley & Sons.

Jordan, Thomas. 1980. *A Handbook of Gravity-Flow Water Systems.* London: Intermediate Technology Publications.

Leder, Charles. 2001. *Water Production/Transmission/Distribution System Pre-Feasibility Study for Pojoaque River Basin, NM.* Albuquerque, NM: ASCG Inc.http://www.ose.state.nm.us/water-info/AamodtSettlement/Appendix03.pdf

Manyanhaire, Itai Offat and Taneal Kamuzungu. 2009. "Access to Safe Drinking Water by Rural Communities in Zimbabwe: A Case of Mundenda Village." *Journal of Sustainable Development in Africa* 11: 113-127.

Mogenson, Ulla. 1996. "Engineering Report for Nan Mand Water Supply System." *SOPAC Technical Report* 241.

Nakayama, Y., R.F. Boucher. 1999. *Introduction to Fluid Mechanics.* New York: John Wiley & Sons.

Tawney, Eric. n.d. *Masters International Peace Corp Program Technical Brief #4: Visualization of Construction of a Gravity-Fed Water Supply System.* http://cee.eng.usf.edu/peacecorps/Resources.htm

U.S. Census Bureau. 2005. "Tesuque CDP, New Mexico 2005-2009 American Community Survey 5-year Estimates." American FactFinder Fact Sheet.

Water.org. 2010. "Learn about the water crisis." Accessed February 20, 2011. http://water.org/learn-about-the-water-crisis/facts/

WHO/Unicef Joint Monitoring Program for Water Supply and Sanitation. 2010. "Improved Drinking-Water Sources: Honduras." Accessed February 20, 2011 http://www.wssinfo.org/fileadmin/user$_$upload/resources/HND$_$wat.pdf

Young, D., B. Munson and F. Okiishi. 2001. *Brief Introduction to Fluid Mechanics* New York: John Wiley & Sons.

12.7 Earthquake Resistant Construction

12.7.1 Introduction

In January 2010, a magnitude 7.0 earthquake hit Haiti, claiming over 200,000 lives. Less than two months later, a much larger 8.8 earthquake hit Chile, and fewer than 1000 lives were lost. Why Chile fared so much better than Haiti was not a result of the accuracy of earthquake predictions (seismology). Rather, lives were saves through the design and retrofitting of earthquake resistant buildings using methods from structural dynamics and civil engineering. In this Module, we will introduce several aspects of earthquake resistant construction. We will begin in Section 2 by discussing how the spring mass equation (normally covered in a first undergraduate level ordinary differential equations course) is fundamental within structural dynamics. In Section 3, we generalize the basic equation using systems of ODE's to handle multiple-storied structures and other basic building design configurations. Finally, in Section 4, we recommend resources for those interested in further study.

12.7.2 The Fundamental Equation of Structural Dynamics

Most beginning undergraduate courses in ordinary differential equations study second order equations with constant coefficients; that is, equations of the form

$$m\,\ddot{u} + c\,\dot{u} + ku = F(t),$$

where m, c and k are constants and the function $F(t)$ on the right hand side is sometimes called the **forcing term**. The dot indicates differentiation with respect to time t. A standard application of this equation is the so called **spring mass problem**, where $u(t)$ indicates the position of a mass m attached to a spring with stiffness constant k. The constant c is called the **damping constant** and the forcing term represents an external force acting on the mass such as a magnet. This equation is called the **fundamental equation of structural dynamics (FESD)**.

In the context of analysis of structures undergoing stress created by ground motion due to earthquakes, $u(t)$ may represent the horizontal motion of the structure, m the mass of the structure, and k the stiffness of the structure (which depends on the material used in construction). The damping term represents the design features of the structure which regard its horizontal displacement. An external force on a lower level of the structure (eg. foundation) may result from horizontal acceleration of an upper level (eg. first floor) which is tied down into the lower level. A force on the upper level might be typically due to wind or perhaps in rare cases due to the impact of an object such as a car or tsunami.

Homogeneous Equation

If there is no external force, FESD is called a **homogeneous equation**:

$$m\,\ddot{u} + c\,\dot{u} + ku = 0.$$

Exact solutions to this equation are well known and depend on the roots $r_{\pm} = \frac{-c \pm \sqrt{c^2 - 4mk}}{2m}$ to the characteristic equation

$$mr^2 + cr + k = 0$$

as shown in the table below. (See for example [Boyce and DiPrima 2009].)

Table 1

$\Delta = c^2 - 4mk$	General Solution
$\Delta > 0$	$u = c_1 e^{r_+ t} + c_2 e^{r_- t}$
$\Delta = 0 (r_+ = r_- = r_0)$	$u = c_1 e^{r_0 t} + c_2 t e^{r_0 t}$
$\Delta < 0 (r_{\pm} = \lambda \pm i\mu)$	$c_1 e^{\lambda t}\cos(\mu t) + c_2 e^{\lambda t}\sin(\mu t)$

Exercise 2.1

Find the general solution to the equation

$$\ddot{u} + c\,\dot{u} + u = 0$$

for each choice of the constant c:

a) $c = 0$. b) $c = 1$. c) $c = 2$. d) $c = 4$.

Damping

For simplicity, let $u(t)$ denote the horizontal displacement at time t from equilibrium of one end of building foundation with unit mass ($m = 1$) and spring constant ($k = 1$) (see figure 1 below).

$u(t)$ gives the horizontal displacement at time t of an elastic object with one end fixed.

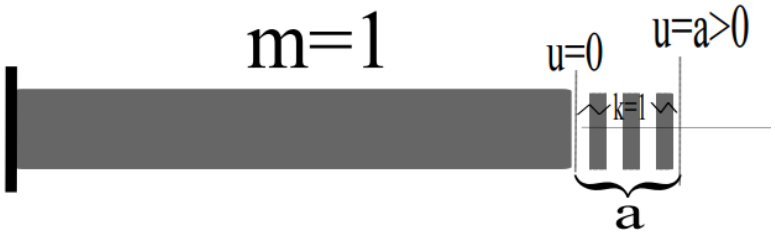


FIGURE 1:

Given the damping coefficient $c \geq 0$ and assuming no external force $F(t)$ acts on the system in the horizontal direction, the equation of motion is

$$\ddot{u} + c\,\dot{u} + u = 0.$$

Assuming that the material is expanded by 1 unit and then released from rest, the initial conditions are $u(0) = 1$, $\dot{u}\,(0) = 0$. Using the solutions to Exercise 2.1, the table below gives solutions to the damped equation with these initial conditions for various values of the damping coefficient c.

c	Solution
0	$u(t) = \cos t$
1	$u(t) = e^{-\frac{1}{2}t}[\cos(\frac{\sqrt{3}}{2}t) + \sin(\frac{\sqrt{3}}{2}t)]$
2	$u(t) = e^{-t}(1+t)$
4	$u(t) = \frac{2+\sqrt{3}}{2\sqrt{3}}\ e^{(-2+\sqrt{3})t} + \frac{-2+\sqrt{3}}{2\sqrt{3}}\ e^{(-2-\sqrt{3})t}$

TABLE 2:

Figure 2 below shows the effect of increasing the damping coefficient c. The vertical axis gives displacement from equilibrium and the horizontal axis represents time.

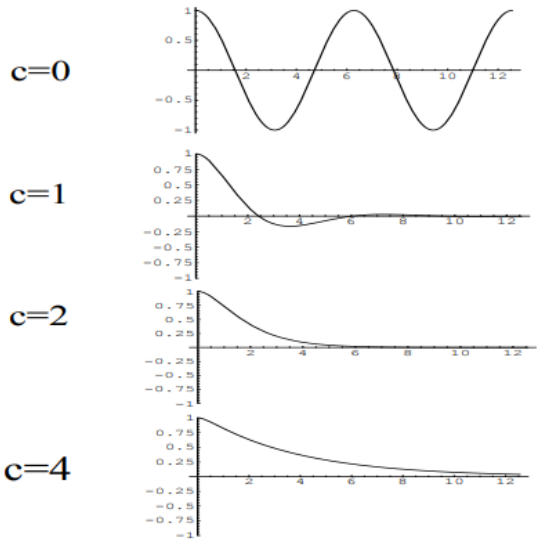


FIGURE 2:

In particular, the effect of the damping may be classified as follows:

$0 \leq c < 2$ underdamped (foundation continues to oscillate indefinitely)

$c = 2$ critical damping (change in behavior occurs)

$c > 2$ overdamped (foundation moves back towards its equilibrium position without oscillating)

In designing earthquake resistant construction, various damping mechanisms have been employed. The more effective the damping, the higher the c value in the equation describing the motion, thereby eliminating horizontal oscillations of the foundation.

Exercise

2.2 Use the general solutions given in Exercise 2.1 to check the solutions in Table 2.

Forcing and Resonance

In the absence of external forces in the direction of motion, the displacement $u(t)$ from equilibrium of a one-degree-of-freedom (1-DOF), undamped ($c = 0$) vibrating media with mass $m > 0$ and spring constant $k > 0$, satisfies the equation

$$m\,\ddot{u} + ku = 0.$$

In this case, the general solution is $c_1 \cos(\sqrt{\frac{k}{m}})t + c_2 \sin(\sqrt{\frac{k}{m}})t$ with *natural frequency* $\omega_0 = \sqrt{\frac{k}{m}}$. Without any damping, we expect the motion to be periodic with period $T_0 = \frac{2\pi}{\omega_0}$

For simplicity, let us assign the parameter values $m = 1$, $k = 4$, and initial conditions $u(0) = 1, \dot{u}\,(0) = 0$. In this case, the solution satisfying these initial conditions is $u(t) = \cos(2t)$ with natural frequency $\omega_0 = 2$ and period $T_0 = \pi$.

Let us now consider the effect of introducing an idealized periodic external force $F(t) = \cos(At)$ (Figure 3) such as might roughly model a quasi-periodic surface shockwave created by an earthquake over a relatively short time span. In this case, our equation becomes

$$\ddot{u} + 4u = \cos(At); \qquad u(0) = 1, \dot{u}\,(0) = 0.$$

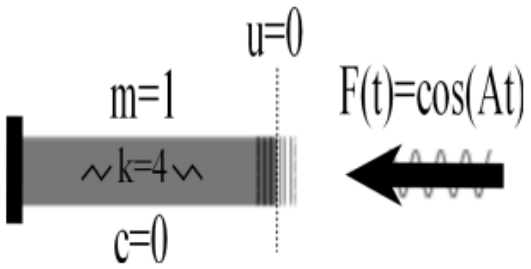


FIGURE 3:

For an equation of the form

$$m\ddot{u} + c\dot{u} + ku = F(t)$$

the general solution has the form $u(t) = u_h(t) + u_p(t)$, where $u_h(t)$ is the general solution to the homogeneous equation ($m\ddot{u} + c\dot{u} + ku = 0$) and $u_p(t)$ is a particular solution to the forced differential. Table 3 gives the form of $u_p(t)$ corresponding to the form of the forcing term $F(t)$ appearing in the equation.

Form of Forcing Term $F(t)$	Form of Particular Solution $u_p(t)$
$(a_0 + a_1 t + ... + a_n t^n)e^{\lambda t}\cos(\mu t)$	$t^\alpha[(A_0 + A_1 t + ... + A_n t^n)e^{\lambda t}\cos(\mu t) +$ $(B_0 + B_1 + ... + B_n t^n)e^{\lambda t}\sin(\mu t)]$ where α is the minimum value from $\{0, 1, 2\}$ that ensures no term in the particular solution satisfies the homogeneous equation.

TABLE 3:

Table 4 gives solutions with these initial conditions for various values of the forcing frequency A Figure 4 shows their graphs.

Forcing frequency A	Equation	Solution
0	$\ddot{u} + 4u = 1$	$u(t) = \frac{1}{4} + \frac{3}{4}\cos(2t)$
1	$\ddot{u} + 4u = \cos t$	$u(t) = \frac{1}{3}\cos t + \frac{2}{3}\cos(2t)$
2	$\ddot{u} + 4u = \cos(2t)$	$u(t) = \frac{1}{4}t\sin(2t) + \cos(2t)$

TABLE 4:

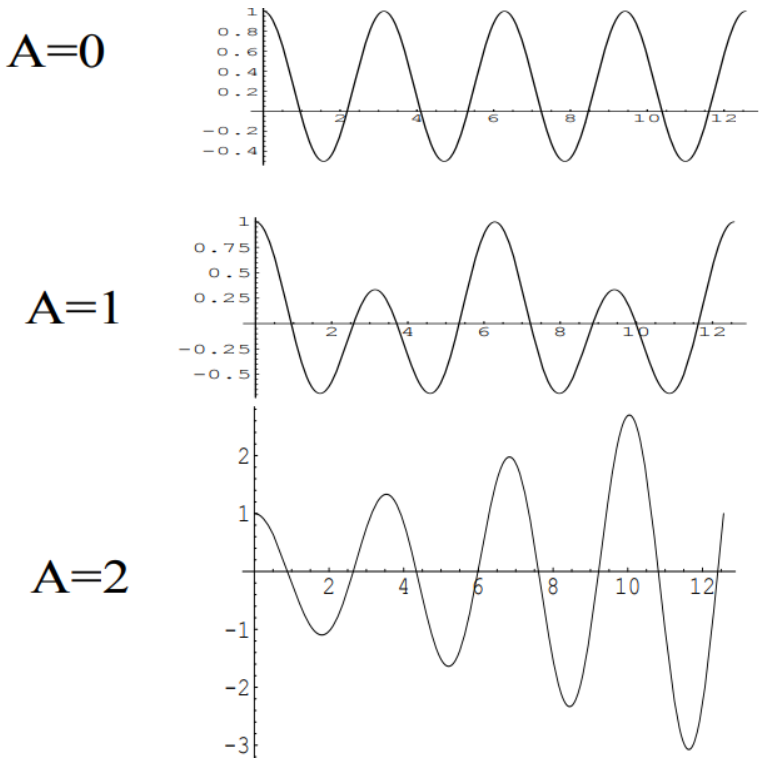


FIGURE 4:

Of particular interest is the solution behavior when the forcing frequency A matches the natural frequency $\omega_0 = 2$. Resonance is the phenomenon that occurs when the frequency of the forcing term matches the natural frequency of the object in question. When this occurs, the amplitude of the oscillations will increase without bound. In the case of a building, the amplitude would increase until structural failure occurs. (A classic example of resonance is the singer who can break a wine glass merely by matching their voice to the natural frequency of the glass).

Exercise

2.3 How does the introduction of damping affect resonance? That is, analyze, the equation

$$\ddot{u} + c\,\dot{u} + 4u = \cos(2t)$$

for small positive c values. Assume that $u(0) = \dot{u}\,(0) = 0$.

Blast Loading

Earlier, we mentioned that a horizontal external force can act on a foundation by transfer of a load exerted on the first floor, such as a force created by a gust of wind.

$$F(t) = -\frac{Fmax}{T}t + Fmax \qquad (0 \le t \le T)$$

$$F(t) = 0 (t > T)$$

where the positive constants F_{max} and T represent the maximum blast force and duration of the blast force respectively.

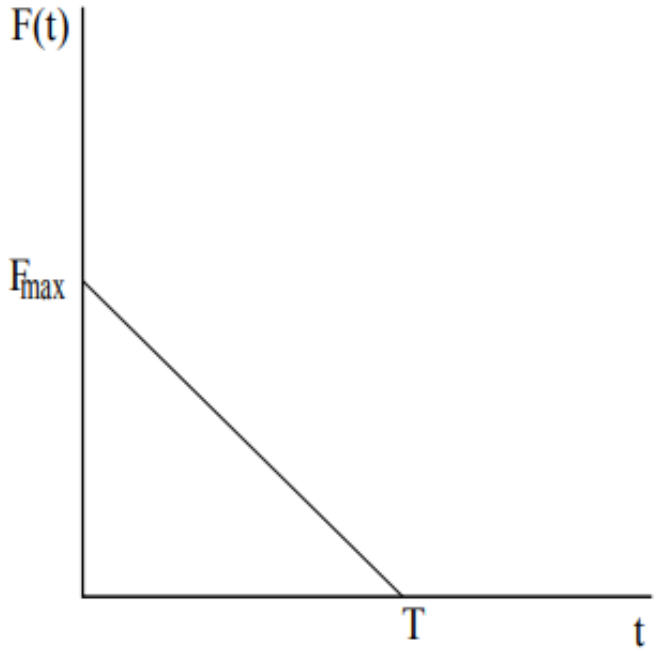


FIGURE 5:

Suppose a building structure with mass $m = 10^4 kg$ is anchored to a foundation in such a way that its resistance to horizontal displacement is modeled by a spring force with constant $k = 10^{10} \frac{kg}{sec^2}$. If a horizontal blast force created by a gust of wind lasts for $T = .5$ sec, and the building can safely undergo a horizontal displacement of 5x10^{-3} meters, what is the maximum blast force $Fmax$ that the building can endure?

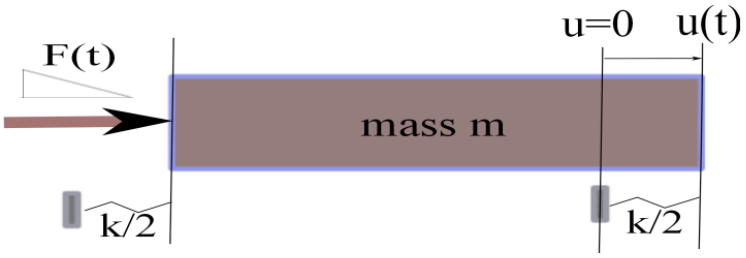


FIGURE 6:

Using the forcing function $F(t)$ given by Figure 5 and initial conditions $u(0) = \dot{u} = 0$, we first obtain $u(t)$ satisfying the differential equation

$$m\,\ddot{u} + ku = F(t)$$

over the time interval $0 \le t \le T$ by means of Table 3 :

$$u(t) = u_1(t) = \frac{Fmax}{k}[1 - \frac{t}{T} - \cos(\omega_0 t) + \frac{\sin \omega_0 t}{\omega_0 T}] \qquad 0 \le t \le T.$$

where $\omega_0 = \sqrt{\frac{k}{m}}$ is the natural frequency. Next, using the initial conditions $u_2(T) = u_1(T) = -\frac{Fmax}{k}$, $\dot{u_2}\ (T) = \dot{u_1}\ (T) = 0$, we obtain the solution for all positive t:

$$u(t) = u_1(t) = \frac{Fmax}{k}[1 - \frac{t}{T} - \cos(\omega_0 t) + \frac{\sin \omega_0 t}{\omega_0 T}] \qquad (0 \le t \le T)$$

$$u(t) = u_2(t) = \frac{Fmax}{k}[-\cos(\omega_0 t)] \qquad (t > T)$$

For the parameter values given in the problem, instead of finding the maximum value of $u(t)$, we instead maximize the **response function** $R(t) = \frac{ku(t)}{Fmax}$ given by

$$R(t) = 1 - \frac{t}{T} - \cos(1000t) + \frac{1}{250\pi} \qquad (0 \leq t \leq T) \sin 1000t \qquad (t > T)$$

$$R(t) = -\cos(1000t)$$

Note that $u(t)$ and $R(t)$ will have their maximum value at the same time t and the response function has a more convenient range of values on the vertical axis.

Figure 7 shows the maximum value of $R(t)$ is $Rmax = 2$.

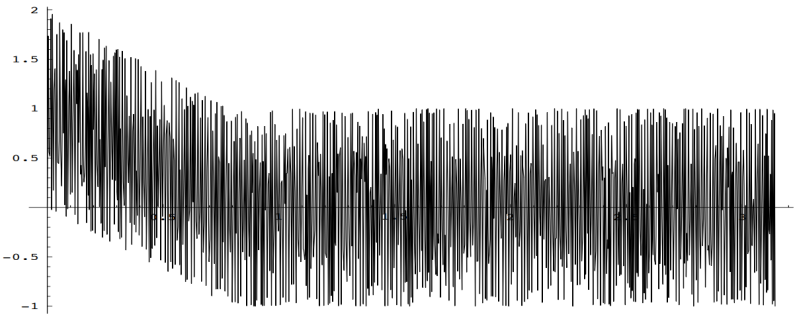


FIGURE 7:

Exercise 2.4 asks you to find the maximum blast force $Fmax$ which the building can endure in answer to our original problem.

Exercise

2.4 Using Figure 7 to estimate the maximum value $Rmax$ of the response function $R(t)$, show that the maximum allowable blast force is $Fmax = 2.5(10^7)N$.

Convolution Integral Method

In this and the next subsection, following Craig and Kurdila [2006], we present two different methods for solving the same blast equation (knowing a variety of methods is important not only to appreciate theoretical creativity in solution methods, but also practically speaking since one method may be much more conducive than another for obtaining a solution to the blast loading equation for a given forcing function $F(t)$).

If an external force $F(t)$ acts over a short time interval $0 \leq t \leq \epsilon$, we define its impulse I by the integral $I = \int_0^\epsilon F(t)dt$. Our standard approach to modeling the horizontal displacement $u(t)$ of an undamped building structure due to an impulse force $F(t)$ is to use the initial value problem

$$m\ddot{u} + ku = F(t), \qquad 0 \leq t \leq \epsilon$$

$$m\ddot{u} + ku = 0, \qquad t > \epsilon$$

$$u(0) = \dot{u}(0) = 0.$$

Alternately, if we integrate the ODE from 0 to ϵ, we obtain

$$m\dot{u}(\epsilon) + k\epsilon\bar{u} = I,$$

where $\bar{u}= \frac{1}{\epsilon}\int_0^{\epsilon} u(t)dt$ is the average horizontal displacement on the time interval $0 \leq t \leq \epsilon$. In the limit as $\epsilon \to 0$, the term $k\epsilon\ \bar{u} \to 0$, and so $\dot{u}\ (0) = I/m$. In other words, an impulse force acting over a very short time interval may be modeled by the IVP

$$m\ \ddot{u} + ku = 0, \qquad \dot{u}\ (0) = I/m, u(0) = 0.$$

The solution is called the **impulse response function** $u(t) = \frac{I}{m\omega_0}\sin(\omega_0 t)$ (where $\omega_0 = \sqrt{\frac{k}{m}}$). Setting $I = 1$, we obtain the **unit impulse response function** $h(t) = \frac{\sin(\omega_0 t)}{m\omega_0}$.

In a similar way, if an impulsive force $dI = F(\tau)d\tau$ acts over a short time interval $\tau < t < \tau + d\tau$, the impulse response of the system is given by $du(\tau) = \frac{dI}{m\omega_0}\sin(\omega_0(t-\tau)) = \frac{1}{m\omega_0}F(\tau)\sin(\omega_0(t-\tau))d\tau$. We may think of an external force F as being a sequence of impulsive forces acting over small sub-intervals comprising a time interval [0,t]. If we sum up the impulse responses $du(\tau)$ as τ goes from 0 to t, we obtain the response $u(t)$ as given by the *convolution integral*.

$$u(t) = \frac{1}{m\omega_0}\int_0^t F(\tau)\sin(\omega_0(t-\tau))d\tau = \int_0^t F(\tau)h(t-\tau)d\tau.$$

Applying the Convolution Integral method to solve the triangular blast loading problem, we compute the following convolution integrals to handle the two cases $0 \leq t \leq T$ and $t > T$:

If $0 \leq t \leq T$, $F(t) = Fmax(\frac{-t}{T} + 1)$, and hence

$$u(t) = u_1(t) = \frac{1}{m\omega_0}\int_0^t F(\tau)\sin(\omega_0(t-\tau))d\tau$$

$$= \frac{Fmax}{m\omega_0}\int_0^t (-\frac{\tau}{T} + 1)\sin(\omega_0(t-\tau))d\tau$$

$$= \frac{Fmax}{m\omega_0}[(\frac{-\tau}{T} + 1)\frac{\cos(\omega_0(t-\tau))}{\omega_0}\ |_0^t - \int_0^t \frac{\cos(\omega_0(t-\tau))}{\omega_0}(\frac{-1}{T})d\tau]$$

$$= \frac{Fmax}{m\omega_0^2}[(\frac{-\tau}{T} + 1)\cos(\omega_0(t-\tau))\ |_0^t - (\frac{\sin(\omega_0(t-\tau)}{\omega_0 T})\ |_0^t]$$

$$= \frac{Fmax}{k}[1 - \frac{t}{T} - \cos(\omega_0 t) + \frac{\sin\omega_0 t}{\omega_0 T}].$$

The case $t > T$ is left to the reader as Exercise 2.4.1.

Exercise

2.4.1 Apply the Convolution Integral method to solve the triangular blast loading problem for the case $t > T$.

The Laplace Transform Method

Laplace Transforms provide a third way to solve the triangular blast loading problem. First, recall the definition

$$\mathcal{L}[u(t)] \stackrel{def}{=} \int_0^{\infty} e^{-st}u(t)dt = U(s).$$

To solve the fundamental IVP in structural dynamics, we need the formulae

$$\mathcal{L}[\dot{u}\ (t)] = s\mathcal{L}[u(t)] - u(0) = sU(s) - u(0),$$

and

$$\mathcal{L}[\ddot{u}\ (t)] = s^2\mathcal{L}[u(t)] - su(0) - \dot{u}\ (0) = s^2U(s) - su(0) - \dot{u}\ (0).$$

Applying these to the fundamental IVP with $u(0) = A$ and $\dot{u}\ (0) = B$ gives

$$L[F(t)] \overset{def}{=} Z(s) = m(s^2U(s) - sA - B) + c(sU(s) - B) + kU(s)$$

where

$$U(s) \overset{def}{=} L[u(t)] = \frac{Z(s) + msA + mB + cB}{ms^2 + cs + k}$$

It follows that

$$u(t) = \mathcal{L}^{-1}[\frac{Z(s) + msA + mB + cB}{ms^2 + cs + k}].$$

For simple cases, obtaining the solution $u(t)$ by finding the inverse transform may require basic algebraic techniques such as completing squares and partial fractions, together with a table of Laplace Transforms such as shown in **Table 5**.

Function	Transform	Notes:
$f(t)$	$L(f(t)) = F(s)$	$L(f) = \int_0^\infty e^{-st} f(t)\, dt$
$f'(t)$	$s\,L(f) - f(0)$	
$f''(t)$	$s^2\,L(f) - s\,f(0) - f'(0)$	
1	$\frac{1}{s}\quad s > 0$	
e^{-at}	$\frac{1}{s+a}\quad s > -a$	
$sin(at)$	$\frac{a}{s^2+a^2}\quad s > 0$	
$cos(at)$	$\frac{s}{s^2+a^2}\quad s > 0$	
$-t\,f(t)$	$F'(s)$	LEMMA: Assuming sufficient smoothness of the function $u(t)$, $\frac{d}{ds}\int_0^\infty u(t)\,e^{-st}\,dt = -\int_0^\infty t\,u(t)\,e^{-st}\,dt$
$t^k\quad k > -1$	$\frac{k!}{s^{k+1}}\quad s > 0$	
$e^{at}f(t)$	$F(s-a)$	
$U_c(t)$	$\frac{e^{-cs}}{s}\quad s > 0$	$U_c(t) = \begin{cases} 0 & \text{if } 0 \le t < c \\ 1 & \text{if } t \ge c. \end{cases}$
$U_c(t)\,f(t-c)$	$e^{-cs}F(s)$	
$\delta(t-c)$	$e^{-sc}\quad c \ge 0$	Delta function satisfies $\int_a^b g(t)\,\delta(t-c)\,dt = g(c)$ if $a \le c \le b$

In Exercise 2.4.2 you are guided through the steps which apply the Laplace Transform Method to solve our triangular blast loading problem.

Exercise

2.4.2 a) First, writing the blast forcing function in the form $F(t) = (1 - U_T(t))(\frac{-(t-T)}{T})$, show that $L[u] = \frac{F_{\max}}{m}\{\frac{-1}{(Ts^2)(s^2+\omega_0^2)} + \frac{1}{s(s^2+\omega_0^2)} + \frac{e^{-Ts}}{Ts^2(s^2+\omega_0^2)}\}$.

b) Show that the transform in part {\bf a)} can be re-written as $L[u] = \frac{F_{\max}}{mT\omega_0^2}\{L[T - t - T\cos(\omega_0 t) + \frac{1}{\omega_0}\sin(\omega_0 t) + U_T(t)[t - T - \frac{1}{\omega_0}\sin(\omega_0(t-T))]]\}$.

c) Obtain $u(t)$ by taking the inverse transform.

Ground Motion

Suppose now in addition to the horizontal motion $u(t)$ caused by a wind force acting on an undamped building ($c = 0$) with spring constant k, the ground upon which the building stands may have horizontal motion $z(t)$ such as might be caused by an earthquake..

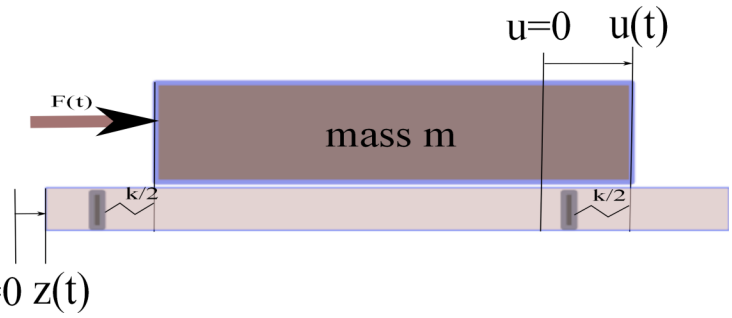


FIGURE 8:

Let $w(t) = u(t) - z(t)$ denote the *relative motion* of the building relative to the ground. Without ground motion, our fundamental equation can be written in the form

$$m\ \ddot{u} = F(t) - ku.$$

With the influence of ground motion, this equation is modified to

$$m\ \ddot{u} = F(t) - k(u - z) \Rightarrow m(\ddot{w} + \ddot{z}) = F(t) - kw(t).$$

Rearranging terms, we obtain

$$m\ \ddot{w} + kw = F(t) - m\ \ddot{z}\ .$$

Exercise

2.5 How does ground motion $z(t)$ effect the fundamental equation of motion?

12.7.3 Two Degree of Freedom Models

Undamped Two-Story Building with Ground Motion

To this point, we have studied **single degree of freedom** models, meaning a single function (scalar differential equation) was adequate to describe the system behavior (namely, horizontal displacement of a building structure). In the case where ground motion occurs, we noted it is useful to model the relative motion $w(t) = u(t) - z(t)$ and consider the ground motion as effectively contributing an inertial force $-m\ \ddot{z}$. In this section we describe a **two-degree of freedom** (2-DOF) model, in which one wind force $F_1(t)$ acts on the first floor of a building with horizontal displacement $u_1(t)$ and a second wind force $F_2(t)$ acts on the second floor of the building with horizontal displacement $u_2(t)$. We also assume an earthquake creates horizontal ground motion $z(t)$.

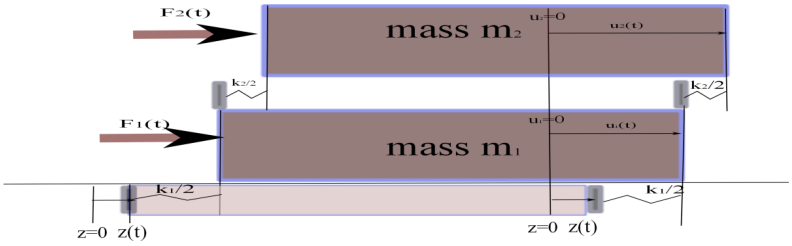


FIGURE 9:

The following two principles are used to construct the equations of motion:

(i) motion of a floor directly below another floor effectively creates a reverse inertial force on the upper floor (Exercise 2.5); and

(ii) if a building maintains its structural integrity (floors do not separate from each other), forces exerted by adjacent floors on each other must be equal in magnitude and opposite in direction;

Applying these principles, we obtain the following system of differential equations for the relative displacements $w_1(t) = u_1(t) - z(t)$ and $w_2(t) = u_2(t) - z(t)$:

$$m_2\ \ddot{u_2} = F_2 - k_2(u_2 - u_1)$$

$$m_1\ \ddot{u_1} = F_1 - k_1(u_1 - z) + k_2(u_2 - u_1)$$

Observing that $u_2 - u_1 = w_2 - w_1$ and $m_i\ \ddot{w_i} = m_i\ \ddot{u_i} - m_i\ \ddot{z}$ we obtain

$$m_2\ \ddot{w_2} = F_2 - k_2(w_2 - w_1) - m_2\ \ddot{z}$$

$$m_1\ \ddot{w_1} = F_1 - k_1(w_1) + k_2(w_2 - w_1) - m_1\ \ddot{z}\ .$$

This system can be written as,

$$M\ \ddot{W} + KW = F$$

where $W = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$ represents the relative displacements, $M = \begin{pmatrix} m_1 & 0 \\ 0 & m_2 \end{pmatrix}$ represents the mass of the floors, $K = \begin{pmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 \end{pmatrix}$ represents the spring constants, and $F = \begin{pmatrix} F_1 - m_1\ \ddot{z} \\ F_2 - m_2\ \ddot{z} \end{pmatrix}$ contains the wind force and the inertial force caused by the ground motion.

Exercise

3.1 Set up a three degree-of-freedom (3-DOF) model $M\ \ddot{W} + KW = F$ which governs the respective relative displacements $w_1(t) = u_1(t) - z(t)$, $w_2(t) = u_2(t) - z(t)$, $w_3(t) = u_3(t) - z(t)$ of the first three floors of a building with ground motion $z(t)$ and respective wind forces $F_1(t)$, $F_2(t)$, and $F_3(t)$.

2-DOF Forced Free Vibration

We now illustrate how to solve a 2-DOF system of the form $M\ \ddot{W} + KW = F$, where $W = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$, $M = \begin{pmatrix} m_1 & 0 \\ 0 & m_2 \end{pmatrix}$, $K = \begin{pmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 \end{pmatrix}$, and $F = \begin{pmatrix} F_1 - m_1\ \ddot{z} \\ F_2 - m_2\ \ddot{z} \end{pmatrix}$.

Consider the simple case where $m_1 = 1, m_2 = 2, k_1 = k_2 = 1, F_1 = 1, F_2 = 0$ and $z(t) = t$. We use the following steps (see Craig and Kurdila [2006]) to solve this system:

(i) Multiplying by M^{-1}, we write the equation in the form $\ddot{W} + \widetilde{K}\, W = \widetilde{F}$:

$\left(\begin{array}{cc}1&0\\0&2 \end{array}\right) \left(\begin{array}{c}\stackrel{..}{w_1} \\ \stackrel{..}{w_2}\end{array}\right)

- \left(\begin{array}{cc}2&-1\\-1&1 \end{array}\right) \left(\begin{array}{c}w_1 \\ w_2\end{array}\right) = \left(\begin{array}{c}1 \\ 0\end{array}\right) \Rightarrow $

$ \left(\begin{array}{c}\stackrel{..}{w_1} \\ \stackrel{..}{w_2}\end{array}\right)

- \left(\begin{array}{cc}2&-1\\-1/2&1/2 \end{array}\right) \left(\begin{array}{c}w_1 \\ w_2\end{array}\right) = \left(\begin{array}{c}1 \\ 0\end{array}\right) $.

(ii) Next, we assume the homogeneous equation $\ddot{W}_h + \widetilde{K}\, W_h = 0$: has a solution of the form $W_h = \begin{pmatrix} w_{h_1} \\ w_{h_2} \end{pmatrix} = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix} \cos(\omega t - \alpha)$. Observing that $\ddot{W}_h = \begin{pmatrix} \ddot{w_{h_1}} \\ \ddot{w_{h_2}} \end{pmatrix} = -\omega^2 \begin{pmatrix} A_1 \\ A_2 \end{pmatrix} \cos(\omega t - \alpha) = -\omega^2 W_h$, we have $-\omega^2 W_h + \widetilde{K}\, W_h = 0$ so ω^2 is an eigenvalue of $\widetilde{K}$: $(\widetilde{K} - \omega^2 I) W_h = 0$:

$$\begin{pmatrix} 2-\omega^2 & -1 \\ -1/2 & 1/2-\omega^2 \end{pmatrix} \begin{pmatrix} A_1 \\ A_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

(iii) To find non-trivial eigenvectors $\begin{pmatrix} A_1 \\ A_2 \end{pmatrix}$ (meaning that A_1 and A_2 are not both zero), the determinant of the coefficient matrix, namely $(2-\omega^2)(\frac{1}{2}-\omega^2) - \frac{1}{2}$, must be zero. This gives two values for ω^2, and each has a corresponding eigenvector Ω obtained by setting $A_1 = 1$:

$\omega_1^2 = \frac{5+\sqrt{17}}{4}$, \hspace{.2in} $\Omega_1 = \begin{pmatrix} 1 \\ \frac{3-\sqrt{17}}{4} \end{pmatrix}$ \vspace{.1in}

$\omega_2^2 = \frac{5-\sqrt{17}}{4}$, \hspace{.2in} $\Omega_2 = \begin{pmatrix} 1 \\ \frac{3+\sqrt{17}}{4} \end{pmatrix}$.

\vspace{.1in}

(iv) Let $\Phi = \begin{pmatrix} \Omega_1 & \Omega_2 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ \frac{3-\sqrt{17}}{4} & \frac{3+\sqrt{17}}{4} \end{pmatrix}$ contain these eigenvectors of $\widetilde{K}$.

Returning to the non-homogeneous problem, the transformation $W = \Phi X = \Phi \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix}$ gives $\Phi\, \ddot{X} + \widetilde{K}\, \Phi X = \widetilde{F} \Rightarrow \ddot{X} + \Phi^{-1}\, \widetilde{K}\, \Phi X = \Phi^{-1}\, \widetilde{F}$. Since $\Phi^{-1}\, \widetilde{K}\, \Phi = \begin{pmatrix} \omega_1^2 & 0 \\ 0 & \omega_2^2 \end{pmatrix}$, we obtain the following uncoupled equations for x_1 and x_2:

$$\ddot{x}_1 + \frac{5+\sqrt{17}}{4} x_1 = \frac{3+\sqrt{17}}{2\sqrt{17}}$$

$$\ddot{x}_2 + \frac{5-\sqrt{17}}{4} x_2 = \frac{-3+\sqrt{17}}{2\sqrt{17}}.$$

For given initial conditions (specifying the values of $x_i(0)$ and $\dot{x}_i\ (0)$), we obtain $x_1(t)$ and $x_2(t)$ in the standard way. Finally, W is obtained from $W = \Phi X$.

Exercise

3.2 Solve for $w_1(t)$ and $w_2(t)$ given the initial conditions $w_1(0) = w_2(0) = \dot{w}_1\ (0) = \dot{w}_2\ (0) = 0$.

Solving Undamped 2-DOF Systems

Following Craig and Kurdila [2006], we present a general method for solving a 2-DOF system such as can be used to describe a two-story building subject to wind forces and ground motion, namely

$$M\ddot{W}+KW=F,$$

where $W=\begin{bmatrix} w_1(t) \\ w_2(t) \end{bmatrix}$, $M=\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix}$, $K=\begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix}$ and $F=\begin{bmatrix} F_1(t) \\ F_2(t) \end{bmatrix}$.

First we consider the unforced system to determine the natural frequency of the building. If we assume the building maintains its integrity, both floors will have the same natural (circular) frequency ω, and phase shift α, although the amplitudes may be different. In particular, we set $W=\begin{bmatrix} A_1\cos(\omega_0 t-\alpha) \\ A_2\cos(\omega_0 t-\alpha) \end{bmatrix}$. ω must then solve the eigenvalue problem

$$(K-\omega^2 M)W=0,$$

or in other words, the characteristic equation

$$det[K-\omega^2 M]=0.$$

Solving this characteristic equation, we obtain ω_1 and ω_2 with $\omega_1^2\leq\omega_2^2$. Next, using the equation

$$(k_{11}-\omega_1^2 m_{11})A_1+(k_{12}-\omega_1^2 m_{12})A_2=0$$

$$(k_{11}-\omega_1^2 m_{11})A_1+(k_{12}-\omega_1^2 m_{12})A_2=0$$

we set $A_1=1$ and call the corresponding values of A_2 from the above equations β_1 and β_2, where

$$\beta_1=-\frac{k_{11}-\omega_1^2 m_{11}}{k_{12}-\omega_1^2 m_{12}}$$

$$\beta_2=-\frac{k_{21}-\omega_2^2 m_{21}}{k_{22}-\omega_2^2 m_{22}}.$$

β_1 and β_2 are called natural (real) modes. The general solution is then obtained as

$$W_1=c_1\cos(\omega_1 t-\alpha_1)+c_2\cos(\omega_2 t-\alpha_2)$$

$$W_2=c_1\beta_1\cos(\omega_1 t-\alpha_1)+c_2\beta_2\cos(\omega_2 t-\alpha_2)$$

where the constants $c_1, c_2, \alpha_1, \alpha_2$ are determined from the initial conditions. The mode shapes (i.e. eigenvectors) are defined as

$\Phi_1=\begin{bmatrix} 1 \\ \beta_1 \end{bmatrix}$ and $\Phi_2=\begin{bmatrix} 1 \\ \beta_2 \end{bmatrix}$

The solution can be expressed as

$$W=c_1\Phi_1\cos(\omega_1 t)+B_1\Phi_1\sin(\omega_1 t)+c_2\Phi_2\cos(\omega_2 t)+B_2\Phi_2\sin(\omega_2 t).$$

Exercise

3.3 Suppose $K=\begin{bmatrix} 1000 & -1000 \\ 1000 & 2000 \end{bmatrix}$ and $M=\begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}$. Determine the natural frequencies ω_1 and ω_2 and mode shapes Φ_1 and Φ_2.

One Story Damped-Building with Ground Motion and a Tilting Tower

In this section we describe another 2-DOF model, which, in addition to one wind force acting on a one-story building creating a horizontal displacement $u(t)$, a second wind force effectively causes a vertical tower atop the building to tilt as modeled by a second function $\theta(t)$ which describes the amount of rotation of the tower (See Figure 10).

The same parameters, namely m (mass of the building), k (spring force) and c (damping force) are used for the horizontal building motion $u(t)$. New parameters needed include:

M (mass of the tower);

a (vertical distance from the top of the building to the center of mass of the tower); and

I_G mass moment of inertia of the tower about its center of mass. (A rigid body with angular speed $\alpha = \frac{d\theta}{dt}$ has kinetic energy $\frac{1}{2} I_G \alpha^2$.)

Following Craig and Kurdila [2006], the equations for $u(t)$ and $\theta(t)$ are both derived from Newton's Law $F = ma$:

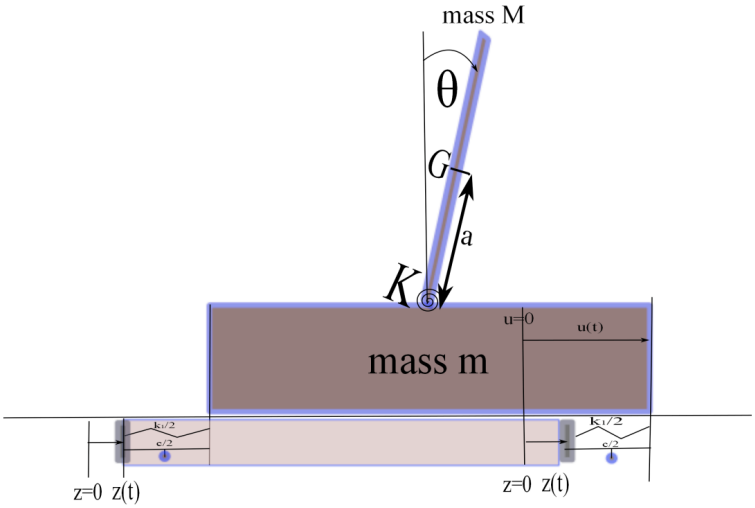


FIGURE 10:

Building: $m\ \ddot{u}\ (t) = -k(u - z) - c(\dot{u} - \dot{z}) - M(\ddot{u} + a\ \ddot{\theta})$

Tower: $I_G\ \ddot{\theta} = -K\theta + Mga\theta - M(\ddot{u} + a\ \ddot{\theta})a.$

Let $U(t) = \begin{pmatrix} u(t) \\ \theta(t) \end{pmatrix}$. T{\flushleft In} **Exercise 3.4** you are asked to show that $U(t)$ satisfies an equation of the form

$$M\ \ddot{U} + C\ \dot{U} + KU = F(t).$$

Note that the form of this differential equation is exactly the same as the fundamental equation!

Exercise

3.4 Identify the components of the matrices U, M, C, K and $F(t)$ in **Figure 10**.

References

Astiz, Luciana and Hiroo Kanamori. ÒAn Earthquake Doublet in Ometepec, Guerrero, Mexico.Ó Physics of the Earth and Planetary Interiors 10 October 1983: 24-45.

Boyce, William E. and R.C. DiPrima. 2009. *Elementary Differential Equations (9E)*, Hoboken, New Jersey: John Wiley & Sons.

Building Seismic Safety Council. 2006. *Homebuilder's Guide to Earthquake Resistant Design and Construction*. Washington D.C.: National Institute of Building Sciences.

Chopra, Anil. 2007. *Dynamics of Structures: Theory and Application of Earthquake Engineering*. Upper Saddle River, NJ: Pearson Prentice Hall.

Craig, Roy R., A Kurdila. 2006. *Fundamentals of Structural Dynamics*. Hoboken, New Jersey: John Wiley & Sons.

Nishenk, S. P. and S. K. Singh. The Acapulco-Ometepec, Mexico, Earthquakes of 1907-1982: Evidence for a Variable Recurrence History.Ó Bulletin of the Seimological Society of America, Vol. 77, No. 4 August 1987: 1359-1367.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint
```

12.8 Solutions to Exercises

12.8.1 Logistic Growth and COVID-19

1) First, separate the variables and antidifferentiate both sides

$$\int \frac{dy}{ky(1-\frac{y}{M})} = \int dt.$$

The partial fraction decomposition

$$\frac{A}{ky} + \frac{B}{1-\frac{y}{M}} = \frac{1}{ky(1-\frac{y}{M})}$$

with $A = 1$ and $B = \frac{1}{kM}$

leads to

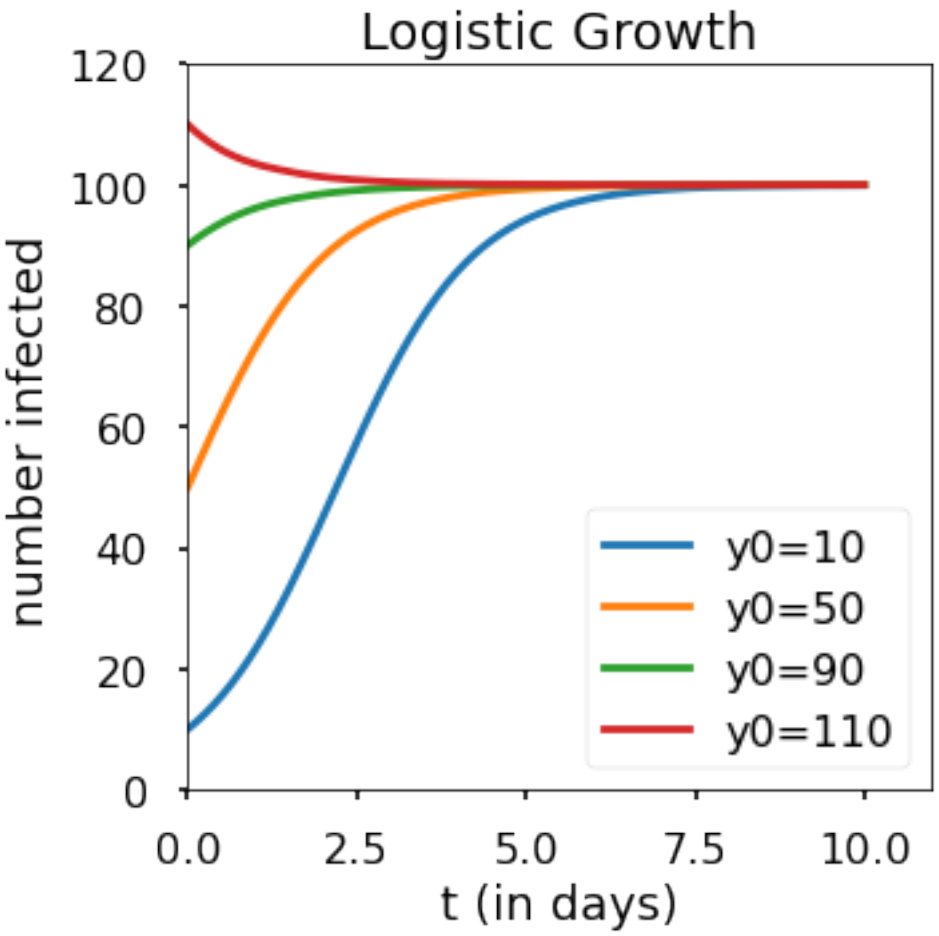
$$\frac{1}{k} \ln \frac{My}{M-y} = t + C.$$

Considering the initial condition $y(0) = y_0 > 0$, the solution can be obtained using algebra:

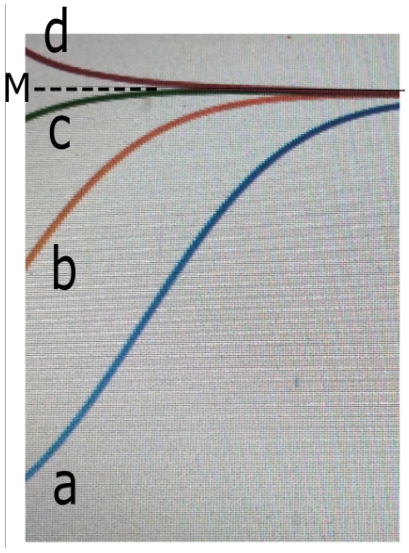
$$y = \frac{M}{1 + (\frac{M}{y_0} - 1)e^{-kt}}$$

2a)

```
<ipython-input-1-b136c51ef48b>:5: MatplotlibDeprecationWarning: The seaborn styles␣
↪shipped by Matplotlib are deprecated since 3.6, as they no longer correspond to␣
↪the styles shipped by seaborn. However, they will remain available as 'seaborn-
↪v0_8-<style>'. Alternatively, directly use the seaborn API instead.
  plt.style.use('seaborn-poster')
```


Logistic Growth
number infected
t (in days)
0
20
40
60
80
100
120
0.0
2.5
5.0
7.5
10.0
y0=10
y0=50
y0=90
y0=110

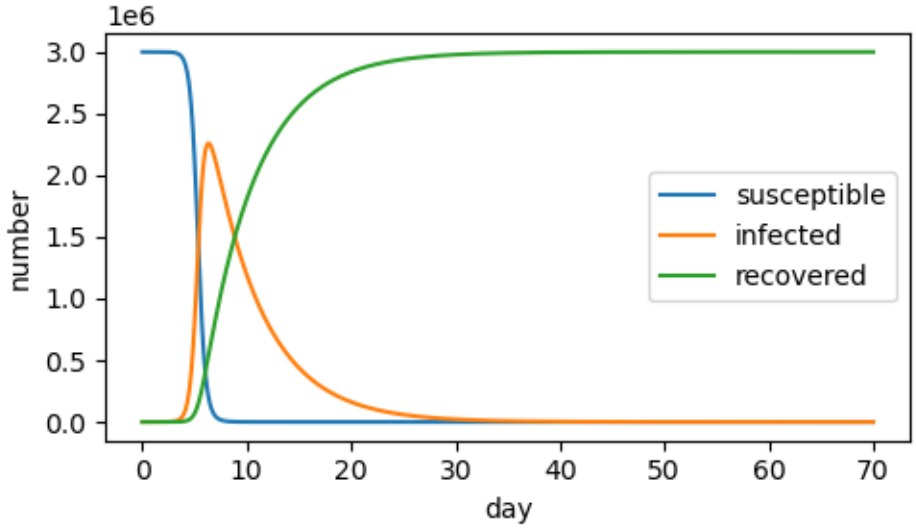
2b)


d
M
c
b
a

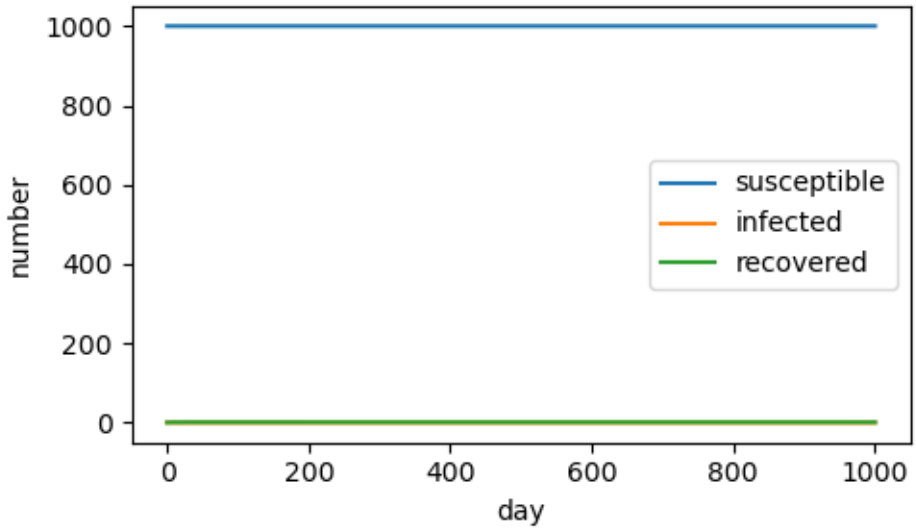
12.8.2 The Basic SIR Model

Exercise 1

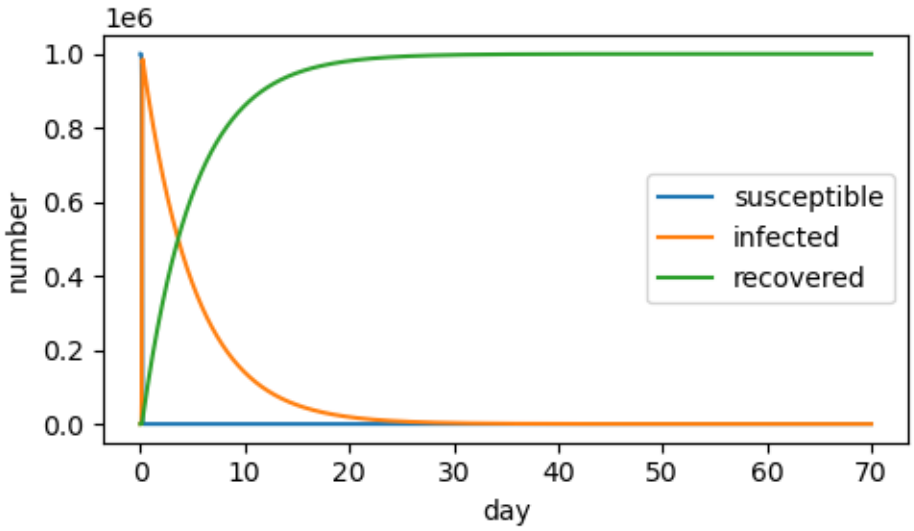
a) If the initial population is increased to 3,000,000, the epidemic appears to end sooner (around day 30)



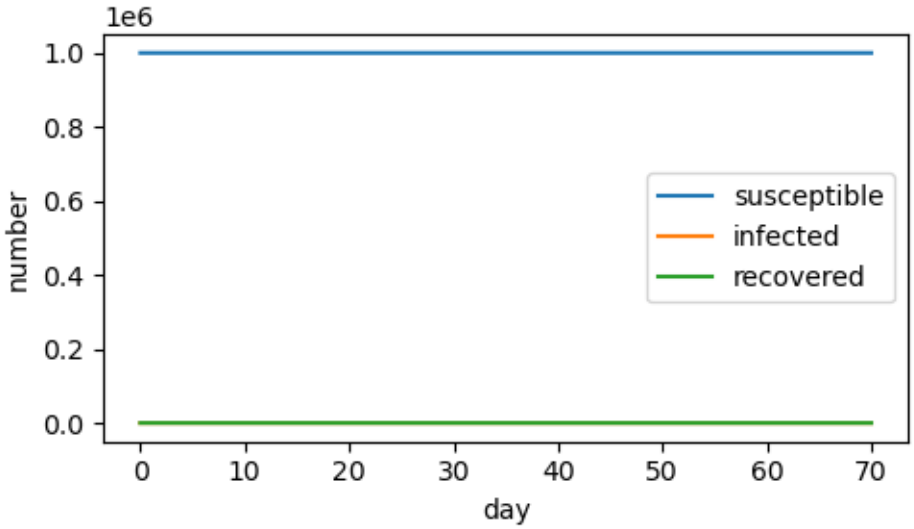
If the initial population is reduced to 1,000, there is no epidemic.



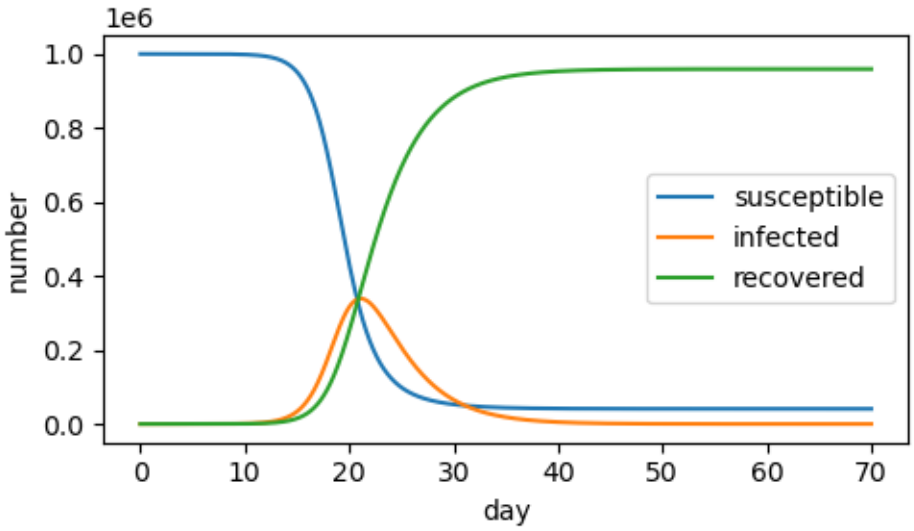
b) If the transmission rate is high ($\beta = .0001$), the entire population is quickly infected, and the epidemic ends sooner (around day 20).



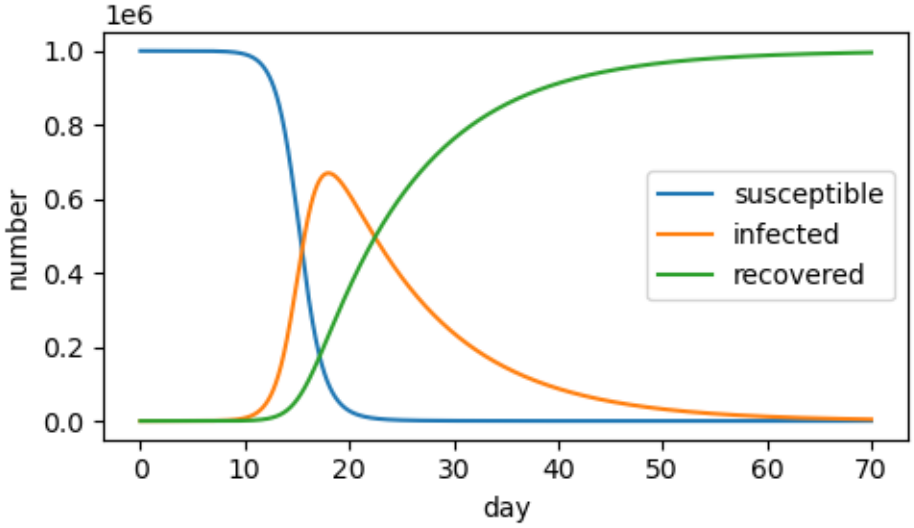
If the infection rate is reduced ($\beta = 10^{-7}$), there is no epidemic.



c) If the recovery rate is increased ($\gamma = .3$), fewer people get infected.



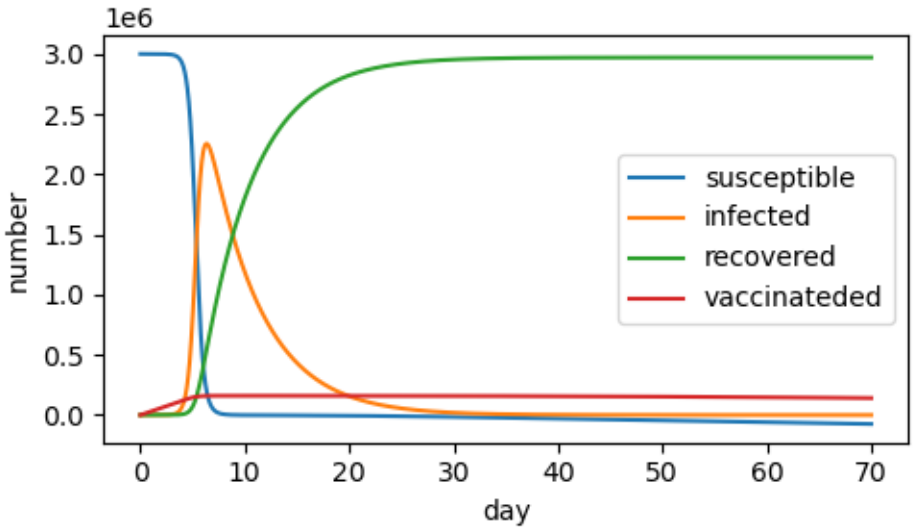
If the recovery rate is decreased ($\gamma = .1$), more people get infected, and the epidemic persists over a longer number of days.

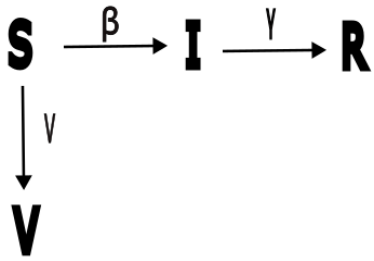


d) As we saw in part b), with a high enough transmission rate, the entire population can become infected.

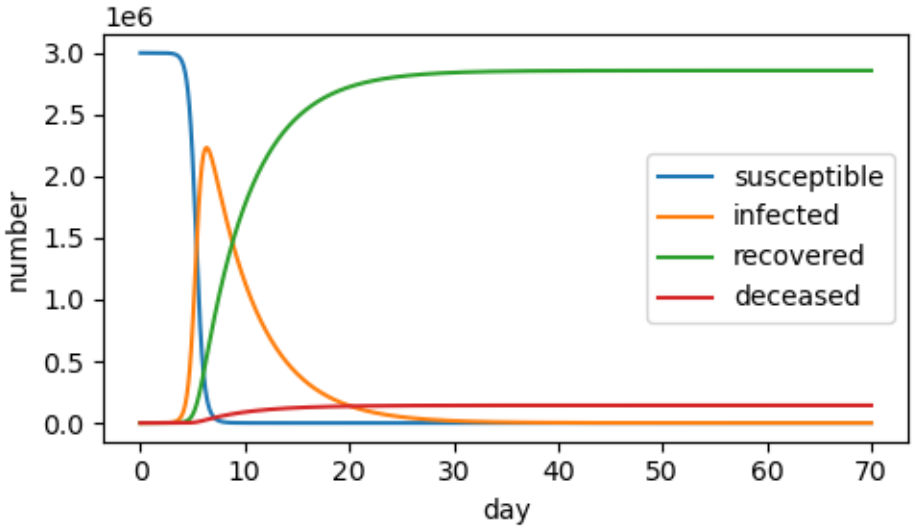
e) As we also saw in part b), with a low enough transmission rate, there is no outbreak.

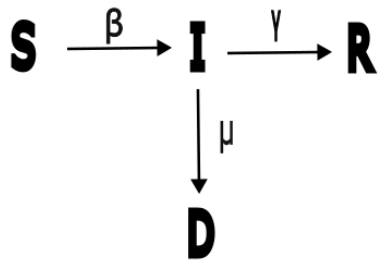
Exercise 2


1e6
3.0
2.5
2.0
1.5
1.0
0.5
0.0
number
0
10
20
30
40
50
60
70
day
susceptible
infected
recovered
vaccinateded


S
β
I
γ
R
ν
V

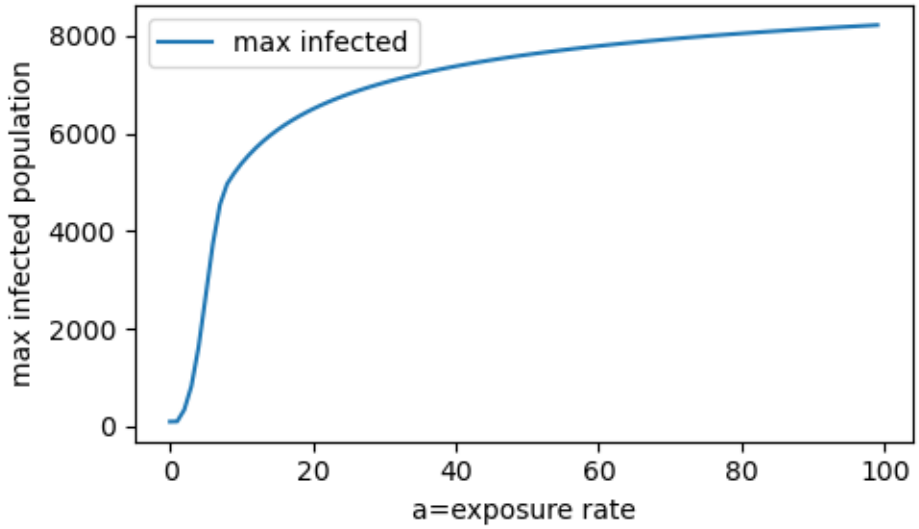
Exercise 3


1e6
3.0
2.5
2.0
1.5
1.0
0.5
0.0
number
0
10
20
30
40
50
60
70
day
susceptible
infected
recovered
deceased



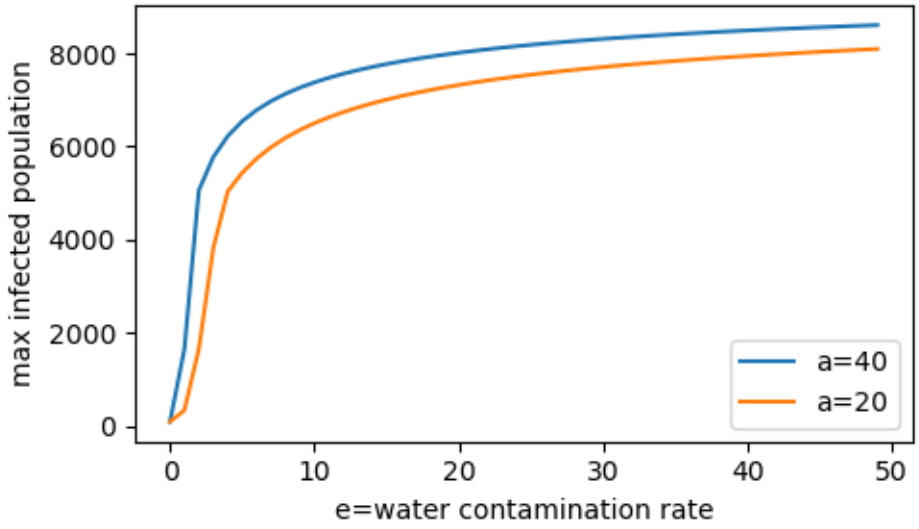
12.8.3 Cholera in Haiti

Exercise 1



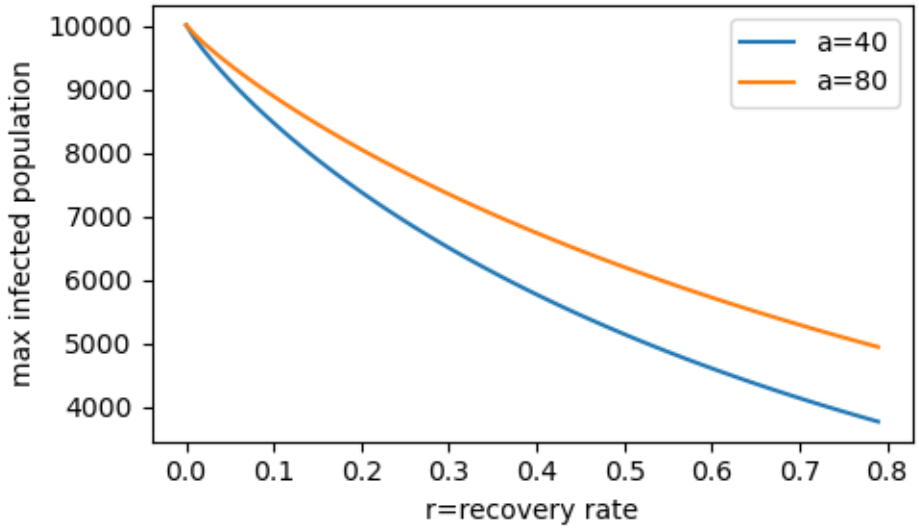
The max infected population rises sharply as the exposure rate (a) is increased from 0 to 10. When a=100, the max_infected population has leveled off at approximately 8000 people.

Exercise 2



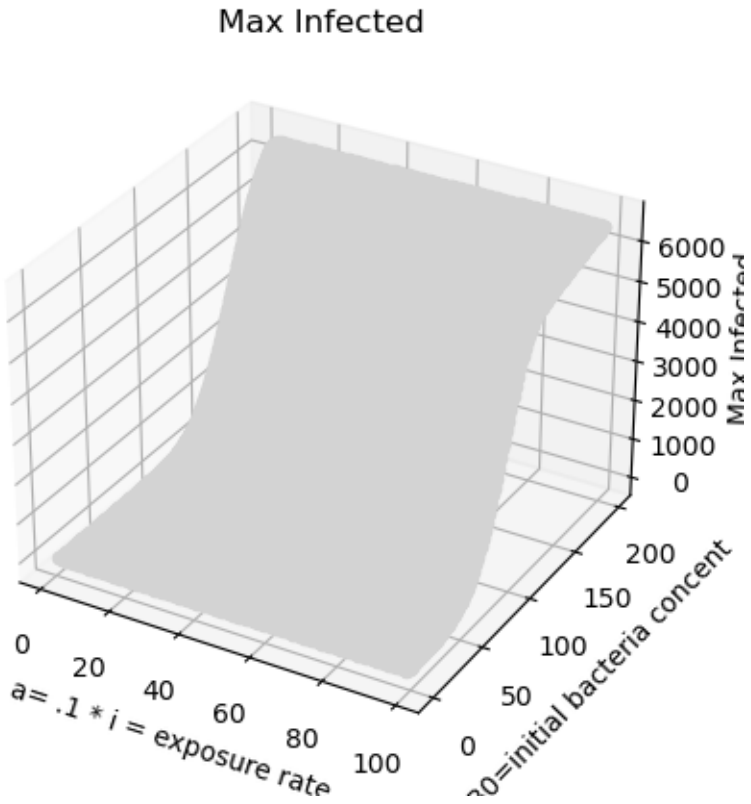
Increased exposure rate (a=20 to a=40) increases the max infected population at each level of the water contamination rate (e).

Exercise 3



Peak infection dependency on the cholera recovery rate r for two exposure levels (a=40/day and a=80/day).

Exercise 4



Exercise 5

a) Though it would be better to revise the model by adding a vaccinated compartment V (t), the basic model might reflect

a vaccination program by lowering the initial susceptible population S(0) and raising the value of K, the latter being the bacteria concentration required to give a .5 probability.

b) Seasonal behavior of cholera outbreaks which may be related to weather events such as floods or El Nino must be considered. That is, even though there may be a period with no or very few cases, there is a latent possibility for re-emergence of a large number of infected population. Mathematically viable solutions may be overly simplistic in light of the reality of everyday life in Haiti.

From the sensitivity analysis, the contamination rate seems to have the largest effect on the infected population, although exposure rate, recovery rate, and initial concentration of bacteria also have an effect. Within the range of 2.5-30 number of people contaminated per day, the infected population will increase significantly. As the contamination rate increases above 30, the infected population stabilizes and increases more slowly. These observations suggest focusing response and preparation measures on lowering the contamination rate

12.8.4 CWS Model of Alzheimer's Disease

Exercise 1

$$(i = 3)\, 0 = Eb_1b_2 + Fb_4 - Eb_1b_3 - Fb_3$$

$$(i = 4)\, 0 = Eb_1b_3 + Fb_5 - Eb_1b_4 - Fb_4$$

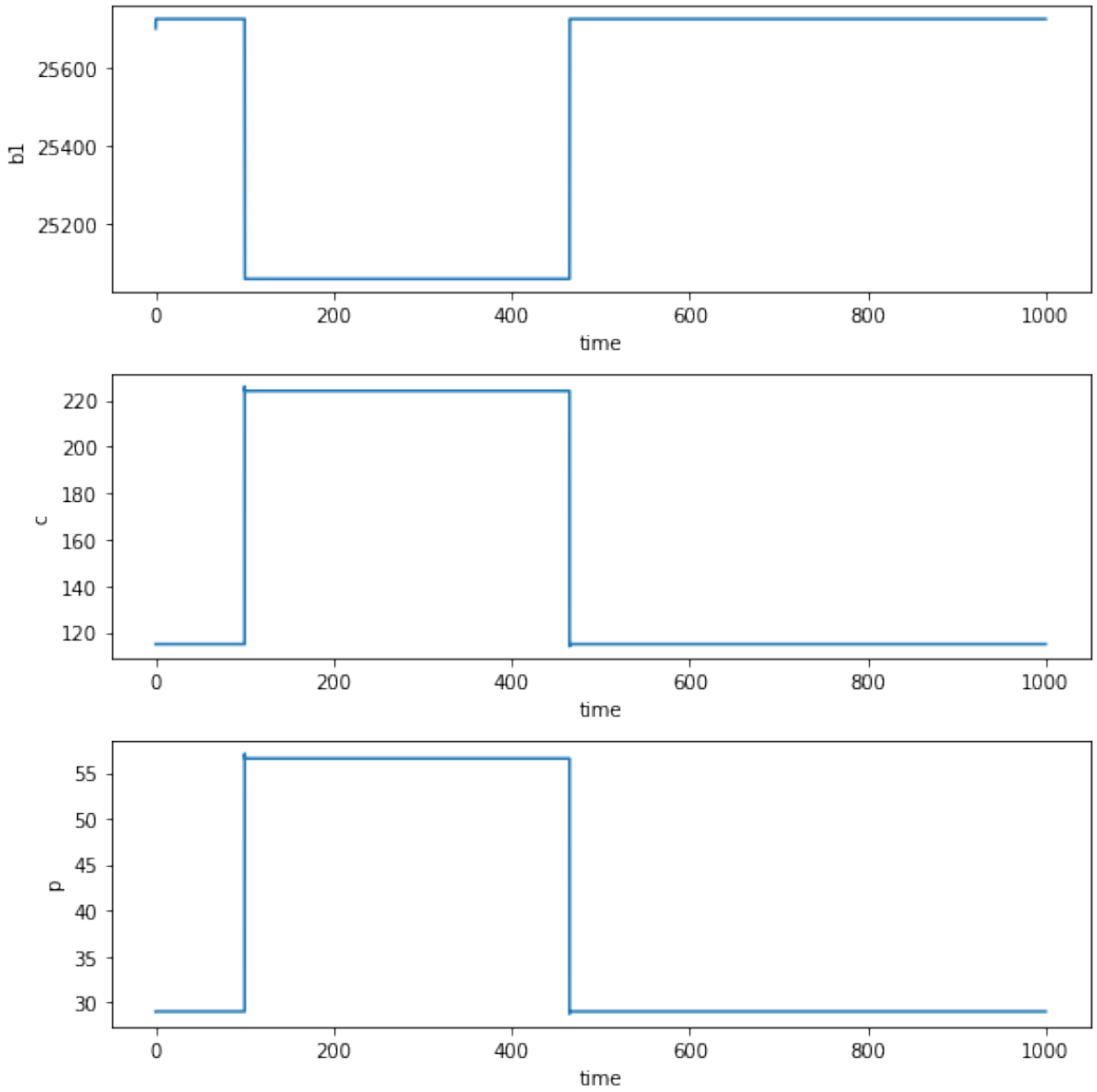
Exercise 2

The total exit rate of monomers from the CSF ($h_2 = r_{21} + r_{23} + L_2$) is the sum of the respective rates monomers move from the CSF to the brain (r_{21} and from CSF to plasma (r_{23}), these being added to the loss of monomers in the brain by degradation (L_2).

Exercises 3

Treatment 2 shows that the plasma and CSF levels of amyloid-beta monomers might increase, but the level in the brain actually decreases.

Exercise 4



The output of the nonlinear is the same as the simplified linear model since the former is equivalent to the latter when $E = S = 0$.

12.8.5 Gravity-Fed Water Delivery

[2.1] $F_{gn} = F_g \cos(\theta) = \gamma \delta V \cos(\theta) = \gamma \delta V \sqrt{1 - (\frac{dz}{ds})^2}$.

[2.2] $\frac{dp}{ds} \approx \frac{p(s_0 - \frac{\delta s}{2}) - p(s_0)}{-\frac{\delta s}{2}} \Rightarrow p(s_0 - \frac{\delta s}{2}) \approx p(s_0) - \frac{dp}{ds}\frac{\delta s}{2}$.

[2.3] Since z has dimension L, we must show the other two terms do as well. Pressure being force per unit area, and force being mass time acceleration, p has dimension $\frac{M\frac{L}{T^2}}{L^2} = \frac{M}{LT^2}$. Since $\gamma = \rho g$ has dimension $\frac{M}{L^3}\frac{L}{T^2} = \frac{M}{L^2T^2}$, the dimension of $\frac{p}{\gamma}$ is $\frac{M}{LT^2}\frac{L^2T^2}{M} = L$. The dimension for $\frac{u^2}{g}$ is $\frac{(\frac{L}{T})^2}{\frac{L}{T^2}} = L$.

[3.1.1]

[a] Bernoulli's equation states $\frac{p}{\gamma}$ + $\frac{u^2}{2g}$ + z = c. Since $u = 0$ $\frac{m}{sec}$ as a result of the tap being closed, $z_B = 0$ m as shown in {\bf Figure \ref{fig9}}, and so our equation simplifies to $\frac{p}{\gamma}$ = c. Observing that c = 9 m + 2 m = 11 ft = $\frac{p\,\frac{kN}{m^2}}{9.789\frac{kN}{m^3}}$, we obtain $p = 105.5\ \frac{kN}{m^2}$.

[b] From part a we know $\frac{p}{\gamma}$ = c. Observing that c = 30 ft + 6 ft = 36 ft = $\frac{p\,\frac{lb}{ft^2}}{62.3\frac{lb}{ft^3}}$, we obtain $p = 2242.8\ \frac{lb}{ft^2} = 2242.8\frac{lb}{ft^2} \cdot \frac{(1ft)^2}{(12in)^2} = 15.6\frac{lb}{in^2}$.

[3.1.2] Referring to Figure (S1), let h be the desired vertical distance from the source reservoir to the faucet. Then h satisfies

$$\frac{1862\frac{kN}{m^2}}{9.789\frac{kN}{m^3}} = h + 3m \Rightarrow h \approx 187m.$$

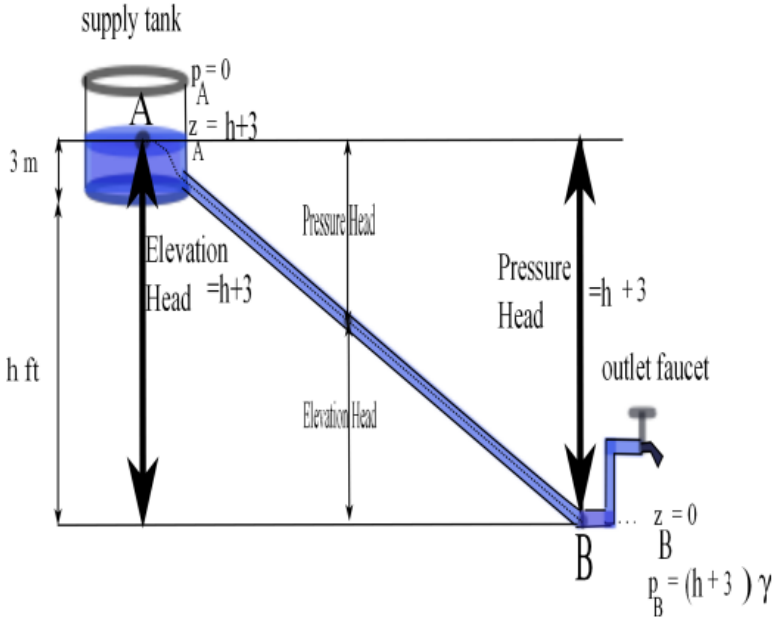


Figure S1. Solution to **Exercise 3.1.2.**

[3.2.1] [a] At a horizontal distance of 600 m, pressure is greatest since the vertical drop from the source is a maximum (114 m).

[b] The pipe can operate at a pressure head of $\frac{p}{\gamma} = \frac{917\frac{kN}{m^2}}{9.789\frac{kN}{m^3}}$ = 93.7 m, so 1 intermediate tank is required.

[3.2.2] Pressure at A is equal to $p_A = [(h_1 + h_2) - (h_3 + h_4)]\gamma$. The pressure at B is equal to $p_B = (h_3 + h_4)\gamma$. The break-pressure tank effectively reduces the pressure at the faucet by the amount p_A.

[3.3.1] .1215 liters/sec.

[3.3.2] [a] A velocity head of 3 m (10 ft) corresponds to a speed $u_1 = [3(2)(9.8)]^{1/2} \approx 7.7$ m/s. By the continuity equation, $\pi(0.025)^2 u_1 = \pi(0.02)^2 u_2$ so $u_2 \approx 11.9$ m/s with a corresponding velocity head of roughly 7.3 m (31.6 feet).

[b]

$$\pi r_1^2 u_0 = \pi r_2^2 u_B \Rightarrow u_0 = (\frac{r_2}{r_1})^2 \sqrt{2g(h_1 + h_2)}.$$

[4.1]

$$\frac{p_1}{\gamma} + \frac{u_1^2}{2g} + z_1 = \frac{p_2}{\gamma} + \frac{u_2^2}{2g} + z_2 + h_f \Rightarrow 0 + 0 + 50 = \frac{p_2}{\gamma} + \frac{2.5^2}{2g} + 30 + 25$$

$$p_2 = (50 - 0.32 - 25 - 30) * 9.789\frac{kN}{m^2} = -52.08\frac{kN}{m^2}$$

A negative pressure means that pressure inside the pipe is less than outside. This can cause dirt or air to be sucked into the pipe, resulting in a stoppage of flow.

[4.2] The Reynolds number $\frac{\rho \bar{u} D}{\mu}$ is dimensionless since all units cancel in the expression $\frac{\frac{M}{L^3}\frac{L}{T}L}{\frac{M}{LT}}$.

[4.3.1] $F_{gs} = -F_g \sin\theta = -mg\sin\theta = -\rho(\pi r^2 L)g\sin\theta = -\gamma\pi r^2 L\sin\theta$.

[4.3.2] $Q = \int_0^{2\pi}\int_0^R u_0(1-(r/R)^2)rdrd\theta = \frac{\pi R^2 u_0}{2}$.

[4.3.3] Since the pipe is wider at B than at A, u_B is less than u_A. Head loss from A to B is represented by h_{AB}.

[4.3.4] $$Re = \frac{\rho\bar{u}D}{\mu} = \frac{998.2\cdot 2\cdot D}{0.001} < 2100 \Rightarrow D < 0.001m = 1mm.$$

The flow rate would be

$$Q = \bar{u}A < \bar{u}\pi(\frac{D}{2})^2 = 1.6x10^{-6}\frac{m^3}{s}.$$

So for a very reasonable $\bar{u}$ the pipe would have to be unreasonably small. This shows that, while laminar flow is a useful assumption to analyze friction, it is unrealistic for gravity-fed water delivery systems.

[4.4.1] The iron pipe has the greater friction factor.

[4.4.2]

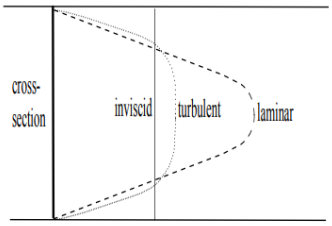


Figure S2. Solution to **Exercise 4.3.2**.

4.5.1. The average velocities associated with the minimum and maximum flow rates are respectively

[4.5.1] The average velocities associated with the minimum and maximum flow rates are respectively $\bar{u}= .35\frac{(10^{-3})}{\pi(.0125)^2} = .71(m/s)$ and $\bar{u}= 1.4\frac{(10^{-3})}{\pi(.0125)^2} = 2.85$ (m/s). The respective Reynolds numbers are $Re = \frac{1.23(.71)(.025)}{1.79(10^{-5})} \approx 1200$, and $Re = \frac{1.23(2.85)(.025)}{1.79(10^{-5})} \approx 4900$. This indicates that the flow at the minimum flow rate is laminar, but will be turbulent at the maximum flow rate. Note in {\bf Table 5} the big difference in head loss between these laminar and turbulent flows: 3.45 m per 100 m at the minimum flow rate versus 49.5 m per 100 m at the maximum flow rate.

[4.5.2] Using the equation for f given in {\bf (\ref{Colebrook})} or a Moody chart , we can determine the Reynolds number Re. We may then compute

$$\bar{u}= \frac{\mu Re}{\rho D},$$

$$Q = \pi(\frac{D}{2})^2\,\bar{u}$$

and

$$h_f = f\frac{L}{D}\frac{\bar{u}^2}{2g}.$$

Knowing the head loss, we can easily plot the HGL on the graph shown in Figure (21). The HGL will help us decide whether the pipe may be used for that segment of the system as explained in Section 4.4.

[5.1] There are different possible designs, but a system using a 80 mm diameter is shown here. In 10 years Tesuque Pueblo will have about 1148 people and a required volumetric flow rate of $Q = 4.27l/s = 0.00427m^3/s$. A 80 mm diameter pipe has a cross-sectional area of 0.005 m^2, so our average velocity is 0.85 m/s (low, but within limits). The maximum static pressure will be $2255 - 1950 = 335$ m at the outlet faucet, which is above the maximum operating pressure head from {\bf Table 5} (146 m). Break pressure tanks at elevations of 2130 m and 2030 m reduce the maximum static pressures to 125 m, 100 m, and 110 m in the first, second, and third sections, respectively (Note that since a 2030 m elevation isn't included in the survey, we don't know if there is a suitable building site at this location. In that case a third break pressure tank would be needed). By rounding our flow rate down we get from {\bf Table 5} a head loss of about 0.96 m per 100 m. We will approximate horizontal distance as pipe length, so our HGL is modeled by the equation $2255 - \frac{0.85^2}{2g} - 0.96x$. If we compare this line to the given elevations we find the elevations remain below the HGL line. This means the dynamic pressure will never be negative and the system is feasible.

12.8.6 Earthquake Resistant Construction

2.1

a) $u = c_1 \cos t + c_2 \sin t$;

b) $u(t) = e^{-\frac{1}{2}t}[c_1 \cos(\frac{\sqrt{3}}{2}t) + c_2 \sin(\frac{\sqrt{3}}{2}t)]$;

c)$u(t) = e^{-t}(c_1 + c_2 t)$;

d) $u(t) = c_1 e^{(-2+\sqrt{3})t} + c_2 e^{(-2-\sqrt{3})t}$.

2.2 For example, to check the solution for $c = 4$, the initial conditions $u(0) = 1$, $\dot{u}\ (0) = 0$ imply that the constants c_1 and c_2 in the general solution $u(t) = c_1 e^{(-2+\sqrt{3})t} + c_2 e^{(-2-\sqrt{3})t}$ must satisfy

$$c_1 + c_2 = 1$$

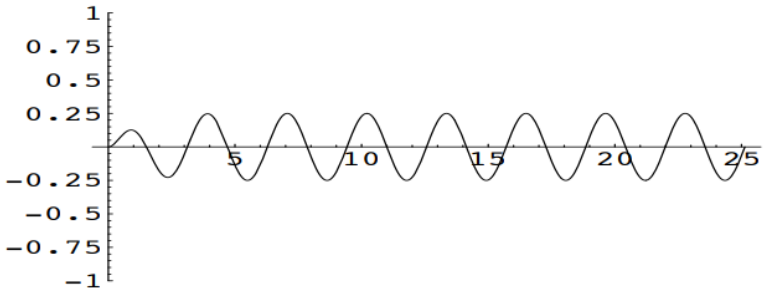
$$c_1(-2 + \sqrt{3}) + c_2(-2 - \sqrt{3}) = 0.$$

It follows easily that $c_1 = \frac{2+\sqrt{3}}{2\sqrt{3}}$ and $c_2 = \frac{-2+\sqrt{3}}{2\sqrt{3}}$.

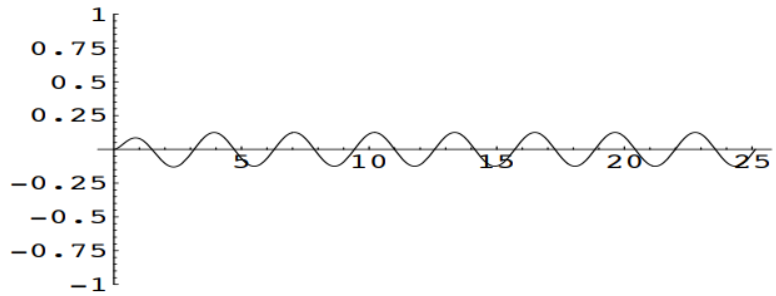
2.3 For $0 < c < 4$, $u = h_h + u_p$ where u_h satisfies

$$\ddot{u_h} + c\,\dot{u_h} + 4u_h = 0,$$

and $u_p = A\cos(2t) + B\sin(2t)$ is a particular solution to the non-homogeneous equation. Solving, we have $u = \frac{1}{2c}\sin(2t) - \frac{2}{c\sqrt{16-c^2}} e^{\frac{-ct}{2}} \sin(\frac{\sqrt{16-c^2}}{2}t)$.



At the critical value $c = 4$, the solution becomes $u = -\frac{1}{4}te^{-2t} + \frac{1}{8}\sin(2t)$



Note that in general, increasing the damping coefficient c will reduce the amplitude of the periodic term $\frac{1}{2c}\sin(2t)$ in the solution and cause a more rapid decay of the transient term $\frac{2}{c\sqrt{16-c^2}}e^{\frac{-ct}{2}}\sin(\frac{\sqrt{16-c^2}}{2}t)$ in the solution.

2.4 Since $Rmax = 2 = \frac{kumax}{Fmax}$ with $k = 10^{10}\frac{kg}{sec}$ and $umax = 5\text{x}10^{-3}$, we obtain the maximum allowable blast force $Fmax = 2.5(10^7)N$.

2.4.1

$$u(t) = u_2(t) = \frac{1}{m\omega_0}\int_0^t F(\tau)\sin(\omega_0(t-\tau))d\tau$$

$$= \frac{1}{m\omega_0}\int_0^T F(\tau)\sin(\omega_0(t-\tau))d\tau + \frac{1}{m\omega_0}\int_T^t F(\tau)\sin(\omega_0(t-\tau))d\tau$$

$$= \frac{Fmax}{m\omega_0}\int_0^T (-\frac{\tau}{T}+1)\sin(\omega_0(t-\tau))d\tau$$

$$= \frac{Fmax}{m\omega_0}[(\frac{-\tau}{T}+1)\frac{\cos(\omega_0(t-\tau))}{\omega_0}\,|_0^T - \int_0^T \frac{\cos(\omega_0(t-\tau))}{\omega_0}(\frac{-1}{T})d\tau]$$

$$= \frac{Fmax}{m\omega_0^2}[(\frac{-\tau}{T}+1)\cos(\omega_0(t-\tau))\,|_0^T - (\frac{\sin(\omega_0(t-\tau)}{\omega_0 T})\,|_0^T]$$

$$\frac{Fmax}{k}[(-\frac{T}{T}+1)(\cos(\omega_0(t-T)) - (-\frac{0}{T}+1)(\cos(\omega_0(t))$$

$$= -\frac{1}{T}\sin(\omega_0(t-T)) + \frac{1}{T}\sin(\omega_0(t))$$

$$= \frac{Fmax}{k}[0 - \cos(\omega_0 t) - \frac{1}{T}\sin(\omega_0(t-T)) + \frac{1}{T}\sin(\omega_0(t))]$$

$$= \frac{Fmax}{k}[-\cos(\omega t)]$$

2.4.2 a)

$$s^2L[u] + \omega_0^2 L[u] = \frac{Fmax}{m}\{L[\frac{-t}{T}+1] - L[(\frac{-t}{T}+1)U_T(t)]\}$$

$$(s^2+\omega_0^2)L[u] = \frac{Fmax}{m}\{\frac{-1}{T}(\frac{1}{s^2}) + \frac{1}{s} - e^{Ts}(\frac{-1}{Ts^2})\}$$

$$L[u] = \frac{Fmax}{m}\{\frac{-1}{(Ts^2)(s^2+\omega_0^2)} + \frac{1}{s(s^2+\omega_0^2)} + \frac{e^{-Ts}}{Ts^2(s^2+\omega_0^2)}\}$$

b) $\$L[u] = \frac{Fmax}{m}\{\frac{\frac{-1}{T}+s}{s^2(s^2+\omega_0^2)} + \frac{\frac{1}{T}+e^{-Ts}}{s^2(s^2+\omega_0^2)}\}\$$

$$L[u] = \frac{Fmax}{mT\omega_0^2}\{\frac{T}{s} - \frac{1}{s^2} - \frac{Ts}{s^2+\omega_0^2} + \frac{1}{s^2+\omega_0^2} + e^{-sT}(\frac{1}{s^2} - \frac{1}{s^2+\omega_0^2})\}$$

$$L[u] = \frac{Fmax}{mT\omega_0^2}\{L[T - t - T\cos(\omega_0 t) + \frac{1}{\omega_0}\sin(\omega_0 t)] + e^{-sT}L[t - \frac{1}{\omega_0}\sin(\omega_0 t)]\}$$

$$L[u] = \frac{Fmax}{mT\omega_0^2}\{L[T - t - T\cos(\omega_0 t) + \frac{1}{\omega_0}\sin(\omega_0 t) + U_T(t)[t - T - \frac{1}{\omega_0}\sin(\omega_0(t-T))]]\}$$

c)

$$u = \frac{Fmax}{mT\omega_0^2}\{T - t - T\cos(\omega_0 t) + \frac{1}{\omega_0}\sin(\omega_0 t) + U_T(t)[t - T - \frac{1}{\omega_0}\sin(\omega_0 t)]\}$$

$$u = \frac{Fmax}{k}\{1 - \frac{t}{T} - \cos(\omega_0 t) + \frac{1}{\omega_0 T}\sin(\omega_0 t) + U_T(t)[\frac{t}{T} - 1 - \frac{1}{\omega_0 T}\sin(\omega_0 t)]\}$$

2.5 The ground motion $z(t)$ effectively creates a reversed inertial force $-m\ \ddot{z}$.

3.1 $W = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix}$ represents the relative displacements, $M = \begin{pmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{pmatrix}$ represents the mass of the floors, $K = \begin{pmatrix} 2(k_1+k_2) & -2k_2 & 0 \\ -2k_2 & 2(k_2+k_3) & -2k_3 \\ 0 & -2k_3 & 2k_3 \end{pmatrix}$ represents the spring constants, and $F = \begin{pmatrix} F_1 - m_1\ \ddot{z} \\ F_2 - m_2\ \ddot{z} \\ F_3 - m_3\ \ddot{z} \end{pmatrix}$ the represents wind force and the inertial force caused by the ground motion.

3.3 $x_1(t) = \frac{6+2\sqrt{17}}{5\sqrt{17}+17} + c_{11}\cos\sqrt{\frac{5+\sqrt{17}}{4}}t + c_{12}\sin\sqrt{\frac{5+\sqrt{17}}{4}}t$ and $x_2(t) = \frac{-6+2\sqrt{17}}{5\sqrt{17}-17} + c_{21}\cos\sqrt{\frac{5-\sqrt{17}}{4}}t + c_{22}\sin\sqrt{\frac{5-\sqrt{17}}{4}}t$. The initial conditions $w_1(0) = w_2(0) = \dot{w_1}\ (0) = \dot{w_2}\ (0) = 0$ imply that $x_1(0) = x_2(0) = \dot{x_1} = (0) = \dot{x_2}\ (0) = 0$, giving $c_{11} = \frac{-6-2\sqrt{17}}{5\sqrt{17}+17}, c_{12} = 0, c_{22} = \frac{6-2\sqrt{17}}{5\sqrt{17}-17}, c_{21} = 0$. Finally, using $W = \Phi X$ gives $w_1(t) = x_1(t) + x_2(t)$, and $w_2(t) = (\frac{3-\sqrt{17}}{4})x_1(t) + (\frac{3+\sqrt{17}}{4})x_2(t)$.

3.4 $U = \begin{bmatrix} u(t) \\ \theta(t) \end{bmatrix}$, $M = \begin{bmatrix} M+m & Ma \\ Ma & I_G + Ma^2 \end{bmatrix}$, $C = \begin{bmatrix} c & 0 \\ 0 & 0 \end{bmatrix}$, $K = \begin{bmatrix} k & 0 \\ 0 & K - Mga \end{bmatrix}$ and $F(t) = \begin{bmatrix} c\ \dot{z} + kz \\ 0 \end{bmatrix}$

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint
from mpl_toolkits import mplot3d
```

12.9 JNB Lab: HIV-AIDS

Acknowledgements

This material was published previously as a UMAP Module titled Immunological and Epidemiological HIV/AIDS Modeling. The UMAP Journal 26(1) (2005) 49-90. Copyright 2005 by COMAP, Inc. All rights reserved. Authors: Paul Isihara with Andrew Barker, Jordan Hostetter, Michael Hubbell, Kyle Hunsberger, Kimberly Waite, and Rebecca Wilson. Permission to make digital or hard copies of part or all of this work for personal or classroom useis granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice. Abstracting with credit is permitted, but copyrights for components of this work owned by others than COMAP must be honored. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior permission from COMAP.

12.9.1 Introduction

The war against HIV/AIDS has been fought for many years on a variety of fronts. Clinical research has shown that early administration of anti-retroviral therapy (ART) can reduce transmission of HIV from one infected partner to another (Cohen et al 2011). Similarly, such therapy can help prevent the transmission of HIV from mother to newborn. Countries such as the African country of Lesotho have experienced having 1/4 of their population HIV-positive. In such situations, there is an urgent need to get the word out to at-risk people to go for HIV testing. Vodafone foundation initiated a mobile outreach program in September 2015, using the mobile platform to register HIV patients and track treatment, as well as send out SMS messages providing information about HIV AIDS, all messages ending with the words "Get tested for HIV". Timely testing can make all the difference.

Mosotho's mother discovered that she was HIV+ early in her pregnancy, and immediately began to take drugs to prevent her from infecting her baby. When Mosotho was born, he was immediately put on the life saving treatment. M'e N'thabang, a mother of eight children, three of whom were HIV+, for months would bring her young grandson, Mosotho, for his pediatric check-up. Today, he is HIV-, a testament to the impact of what proper health care can do for people. (allafrica.com, 2016)

12.9.2 UNAIDS Fact Sheet

Here is the data from a recent UNAIDS fact sheet highlighting how grave the HIV situation is. (https://www.unaids.org/en/resources/fact-sheet)

Global HIV statistics

- 39.0 million [33.1 million–45.7 million] people globally were living with HIV in 2022.
- 1.3 million [1 million–1.7 million] people became newly infected with HIV in 2022.
- 630 000 [480 000–880 000] people died from AIDS-related illnesses in 2022.
- 29.8 million people were accessing antiretroviral therapy in 2022.
- 85.6 million [64.8 million–113.0 million] people have become infected with HIV since the start of the epidemic.
- 40.4 million [32.9 million–51.3 million] people have died from AIDS-related illnesses since the start of the epidemic.

Infected Population

- In 2022, there were 39.0 million [33.1 million–45.7 million] people living with HIV.
  - 37.5 million [31.8 million–43.6 million] adults (15 years or older).
  - 1.5 million [1.2 million–2.1 million] children (0–14 years).
  - 53% of all people living with HIV were women and girls.
  - 86% [73– 98%] of all people living with HIV knew their HIV status in 2022.
- About 5.5 million people did not know that they were living with HIV in 2022.

People living with HIV accessing antiretroviral therapy

- At the end of December 2022, 29.8 million people were accessing antiretroviral therapy, up from 7.7 million in 2010.
- In 2022, 76% [65–89%] of all people living with HIV were accessing treatment.
- 77% [65–90%] of adults aged 15 years and older living with HIV had access to treatment, as did 57% [44–78%] of children aged 0–14 years.
- 82% [69–95%] of women aged 15 years and older had access to treatment; however, just 72% [60–84%] of men aged 15 years and older had access.

- 82% [64–98%] of pregnant women living with HIV had access to antiretroviral medicines to prevent transmission of HIV to their child in 2022.

New HIV infections

- New HIV infections have been reduced by 59% since the peak in 1995.
- In 2022, 1.3 million [1 million–1.7 million] people were newly infected with HIV, compared to 3.2 million [2.5 million–4.3 million] people in 1995.
- Women and girls accounted for 46% of all new infections in 2022.
- Since 2010, new HIV infections have declined by 38%, from 2.1 million [1.6 million–2.8 million] to 1.3 million [1 million–1.7 million] in 2022.
- Since 2010, new HIV infections among children have declined by 58%, from 310 000 [210 000–490 000] in 2010 to 130 000 [90 000–210 000] in 2022.

AIDS-related deaths

- AIDS-related deaths have been reduced by 69% since the peak in 2004 and by 51% since 2010.
- In 2022, around 630 000 [480 000–880 000] people died from AIDS-related illnesses worldwide, compared to 2.0 million [1.5 million–2.8 million] people in 2004 and 1.3 million [970 000–1.8 million] people in 2010.
- AIDS-related mortality has declined by 55% among women and girls and by 47% among men and boys since 2010.

Key populations

Globally, median HIV prevalence among the adult population (ages 15-49) was 0.7%. However, median prevalence was higher among key populations:

- 2.5% among sex workers
- 7.7% among gay men and other men who have sex with men
- 5.0% among people who inject drugs
- 10.3% among transgender persons
- 1.4% among people in prisons.

Women and Girls

- Globally 46% of all new HIV infections were among women and girls (all ages) in 2022.
- In sub-Saharan Africa, women and girls (all ages) accounted for 63% of all new HIV infections. In all other geographical regions, over 70% of new HIV infections in 2022 occurred among men and boys.
- Every week, 4000 adolescent girls and young women aged 15–24 years became infected with HIV globally in 2022. 3100 of these infections occurred in sub-Saharan Africa.

Testing and Treatment Targets (95–95–95)

- In 2022, 86% [73– >98%] of all people living with HIV knew their HIV status.
  - Among people who knew their status, 89% [75– >98%] were accessing treatment.
  - And among people accessing treatment, 93% [79– >98%] were virally suppressed.
  - Among children ages 0-14 years the 95-95-95 targets were 63% [49% - 86%], 91% [71% - >98%], 81% [63%->98%]
  - Among women, the 95-95-95 targets were: 90% [76->98%], 91% [77->98%], and 93% [79->98%].
  - Among men, the 95-95-95 targets were: 83% [70-98%] of adult men living with HIV knew their HIV status, 86% [72->98%] were accessing treatment and 94% [79->98%] were virally suppressed.

- Among all people living with HIV, 86% [73– >98%] knew their status, 76% [65–89%] were accessing treatment and 71% [60–83%] were virally suppressed in 2022.

Investments

- At the end of 2022, 20.8 billion (in constant 2019 United States dollars) was available for the AIDS response in low- and middle-income countries—around 60% was from domestic sources.
- UNAIDS estimates that 29 billion (in constant 2019 United States dollars) will be required for the AIDS response in low- and middle-income countries, including countries formerly considered to be upper-income countries, in 2025 to get on track to end AIDS as a public health threat.

12.9.3 Perelson's Immunological HIV Model

Almost 50 years have passed since the first immunological model of HIV was proposed by Leon Cooper [1986]. Differential equation models and discrete difference equations have been used by both immunologists and epidemiologists in the war against HIV/AIDS. Whereas immunological models describe how the HIV virus attacks the body's immune system, epidemiological models describe the spread of the HIV/AIDS throughout a population. With the understanding that all mathematical models depend on simplifying assumptions which need to be revised as further experimental evidence/data is obtained (Covert and Kischner 2000), Perelson's model, as introduced briefly in this lab, will help us understand why ART helps the human immune system fight HIV (see Isihara et. al 2005 for a more complete introduction).

Our immune system has two lines of defense: *innate* (i.e., general defense against all invaders) and *acquired* (i.e., defense targeted at specific invaders). Our front-line defense keeps invaders from entering the body or bloodstream, and secondary defense fights invaders that have passed through the front-line. Whereas the innate immune system (eg. skin, stomach acid, mucus, phagocytes) works in both lines of defense, the acquired immune system only works in the second-line via lymphocytes which target *antigens*, (i.e. large molecules on the surfaces of cells, viruses, bacteria etc.) which uniquely identify invaders. T-cells are a type of lymphocyte produced in the marrow. $CD4^+$ T-cells normally average about 1000 per cubic mm of blood and a decrease in their concentration level is used to assess HIV positive status. Perelson's model describes the change in concentration of healthy $CD4^+$ T-cells after an individual becomes infected with the HIV virus. After the primary infection, an HIV positive individual may go through a latency period which may vary considerably before the onset of AIDS. During this period, while free HIV viral cells attach themselves to healthy T-cells resulting in latently affected T-cells, the healthy T-cell concentration remains sufficiently high so that the immune system remains functional. AIDS occurs when the T-cell concentration drops and the HIV viral concentration rises sufficiently, thereby crippling the acquired immune system. AIDS patients are then at critical risk to succumb to other diseases such as pneumonia.

Perelson's Immunological Model is a non-linear system of four ordinary differential equations. Perelson's model considers the qualitative dynamics of the healthy T-cell and HIV-viral concentrations shown in Figure 1 below.

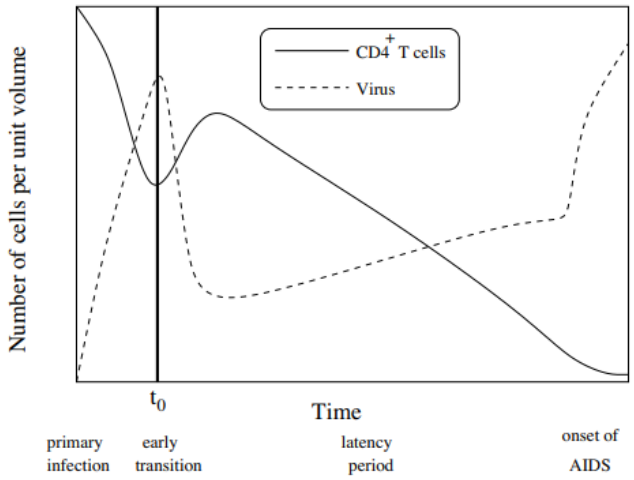
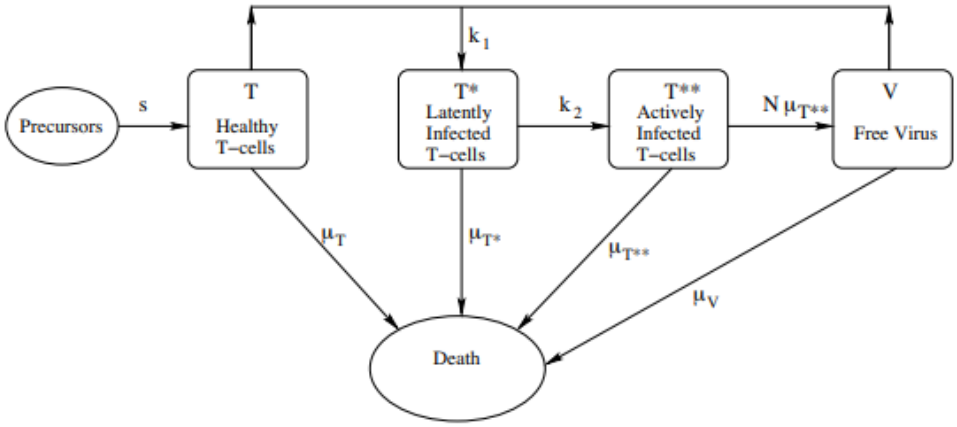


Figure 1. Qualitative dynamics of the healthy T-cell and HIV-viral concentrations based on clinical data. The Perelson immunological model simulates the dynamics beginning at $t = t_0$.

In this model, $T(t)$, $T^*(t)$, $T^{**}(t)$ and $V(t)$ represent respectively the concentrations of healthy T-cells, latently infected T-cells, actively infected T-cells, and HIV viral cells. The four equations specify the dynamics shown in Figure 2.



Change in Healthy T-Cells Equation (12.4)

$$dT/dt = s + rT(1 - \frac{T + T^* + T^{**}}{T_{max}}) - \mu_T T - k_1 TV \tag{12.4}$$

This equation gives the rate of change $\frac{dT}{dt}$ in the concentration $T(t)$ of healthy T-cells New, healthy T-cells enter into the blood stream at a constant rate s (this is an oversimplification, since the rate is expected to decrease during the course of the HIV infection. See Kirschner and Webb 1996). In the absence of free virus (V = 0), the entire right-hand side of the equation reduces to a modified logistic growth equation. The $-k_1 TV$ term assumes that the rate of infection of T-cells by free viral cells is jointly proportional to the concentration of T-cells and the concentration of free virus.

Change in Infected T-Cells Equation (12.5)

$$dT^*/dt = k_1 TV - \mu_T T^* - k_2 T^* \tag{12.5}$$

This equation gives the rate of change $\frac{dT^*}{dt}$ in the concentration $T^*(t)$ of latently infected T-cells. Growth is due to the infection of healthy T-cells with free viral cells ($k_1 TV$). Decreases are due to death ($-\mu_{T^*} T$) (the death rates may be different for healthy, latently infected, and actively infected T-cells), and by progression from latent to active infection ($-k_2 T^*$).

Change in Actively Infected T-Cells Equation (12.6)

$$dT^{**}/dt = k_2 T^* - \mu_b T^{**} \tag{12.6}$$

This equation gives the rate of change $\dfrac{dT^{**}}{dt}$ in the concentration $T^{**}(t)$ of actively infected T-cells. Growth is due to latently infected cells becoming actively infected ($k_2 T^*$). Decrease in concentration is due only to death ($_{T^{**}}T^{**}$).

Change in Virus Concentration Equation (12.7)

$$dV/dt = N\mu_b T^{**} - k_1 TV - \mu_V V \tag{12.7}$$

This equation gives the rate of change $\dfrac{dV}{dt}$ in the concentration $V(t)$ of the free virus. Growth occurs when an actively infected T-cell lyses (i.e., explodes). It is assumed that N copies of the free viral cell are created upon lysing ($N_{T^{**}}T^{**}$). The free viral concentration decreases as the free virus becomes attached to healthy T-cells ($-k_1 TV$) and also through death ($-_V V$).

In particular, Equation HIV1 tells us latently infected T-cells are created when free HIV viral cells attach to healthy T-cells with rate constant k_1. Also, Equation HIV4, giving the rate of change $\dfrac{dV}{dt}$ in the concentration $V(t)$ of the HIV virus indicates that when an actively infected T-cell lyses (i.e. explodes), it releases N copies of the HIV viral cell ($N\mu_b T^{**}$). The free viral concentration decreases as the free virus becomes attached to healthy T-cells ($-k_1 TV$) and also through death ($-\mu_V V$).

12.9.4 Equilibrium Analysis

A key to understanding how Perelson's model simulates the lowering of the healthy T-cell concentration is based on an equilibrium analysis. Full details of this equilibrium analysis can be found here.

The equilibrium points for the system are obtained by solving the following system of algebraic equations:

$$\begin{aligned}
\frac{dT}{dt} = 0 \quad &\Longrightarrow \quad s + rT\left(1 - \frac{T + T^* + T^{**}}{T_{\max}}\right) - \mu_T T - k_1 TV = 0, \\
\frac{dT^*}{dt} = 0 \quad &\Longrightarrow \quad T^* = \frac{k_1}{\mu_{T^*} + k_2} TV, \\
\frac{dT^{**}}{dt} = 0 \quad &\Longrightarrow \quad T^{**} = \frac{k_2}{\mu_{T^{**}}} T^* = \frac{k_2 k_1}{\mu_{T^{**}}(\mu_{T^*} + k_2)} TV, \\
\frac{dV}{dt} = 0 \quad &\Longrightarrow \quad N\mu_{T^{**}} T^{**} - k_1 TV - \mu_V V = 0, \\
&\Longrightarrow \quad \left[\left(\frac{Nk_2 k_1}{\mu_{T^*} + k_2} - k_1\right) T - \mu_V\right] V = 0.
\end{aligned}$$

Using the parameter values shown in Table 1 below, the system has two different equilibrium points for the healthy T cell concentration. Figure 3 shows that for lower values of the lysing parameter N, the equilibrium value corresponding to a healthy person's T-cell concentration is stable.

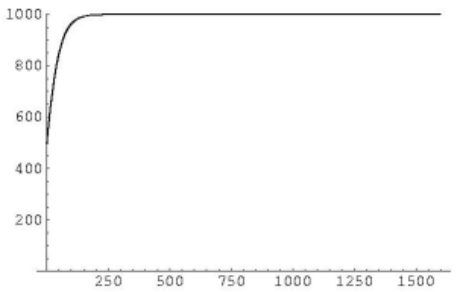


Figure 3. $T(t)$ converges to the stable equilibrium value $T_{\text{uninfected}} = 1000$ when $N = 500$.

Figure 4 shows at higher values of N a loss of stability of the equilibrium representing healthy T-cell concentrations and gain of stability of the other equilibrium representing a low T-cell count characteristic of the onset of AIDS.

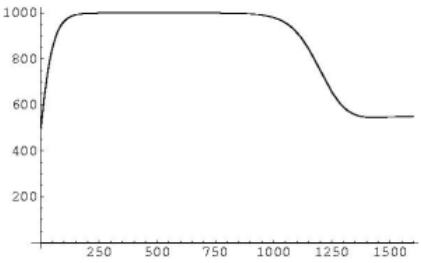


Figure 4. When $N = 1400$, the equilibrium value $T_{\text{uninfected}} = 1000$ is unstable but the equilibrium value $T_{\text{uninfected}} = 580$ is stable.

Figure 5 shows the long term stability of T-cells as N grows larger and larger, displaying

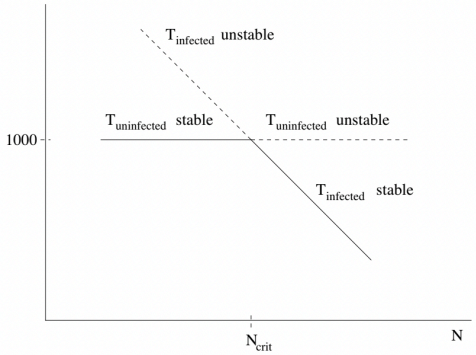


Figure 5. In the Perelson immunological model, a transcritical bifurcation occurs in which the two equilibrium solutions with $T = T_{\text{uninfected}}$ and $T = T_{\text{infected}}$ exchange stability as the parameter N crosses a critical value $N_{\text{crit}} \approx 774$.

One type of anti-retrovial drug called a nucleotide reverse transcriptase inhibitor (NRTI), for example AZT, reduces the lysing number N by preventing the virus from reproducing inside infected T-cells.

12.9.5 Analysis in Python

The script below can be used to solve the Perelson model numerically using the initial and constant values given in Table 1.

Table 1.
Variables and parameters in Perelson's immunological model **(21)–(24)**.

		Initial or default value
Independent Variable		
t	time	days
Dependent Variables		
T	Uninfected CD4$^+$ cell concentration	500 mm^{-3}
T^*	Latently infected CD4$^+$ helper cell concentration	0 mm^{-3}
T^{**}	Actively infected CD4$^+$ helper cell concentration	0 mm^{-3}
V	Free HIV viral concentration	10^{-3} mm^{-3}
Parameters and constants		
s	Rate of supply of CD4$^+$ T cells from precursors	10 mm^{-3} day^{-1}
r	Growth rate constant for the CD4$^+$ T cells	0.03 day^{-1}
$T_{\max}$	Maximum CD4$^+$ T cell concentration	1500 mm^{-3}
μ_T, μ_{T^*}	Death rates of uninfected and latently infected CD4$^+$ T cells	0.02 day^{-1}
$\mu_{T^{**}}$	Death rate of actively infected CD4$^+$ T cell population	0.24 day^{-1}
μ_V	Death rate of free virus	2.4 day^{-1}
k_1	Rate constant for infection of CD4$^+$ T cells with free virus	2.4×10^{-5} mm^3 day^{-1}
k_2	Rate latently infected CD4$^+$ T cells convert to actively infected CD4$^+$ T cells	3×10^{-3} day^{-1}
N	Number of free virus produced by lysing a CD4$^+$ T cell	varies
Derived quantities		
$T_{\text{uninfected}}$	Steady-state concentration of CD4$^+$ T cells in uninfected individuals	1000 mm^{-3}
N_{crit}	Critical number of viral progeny needed for endemic infection	774

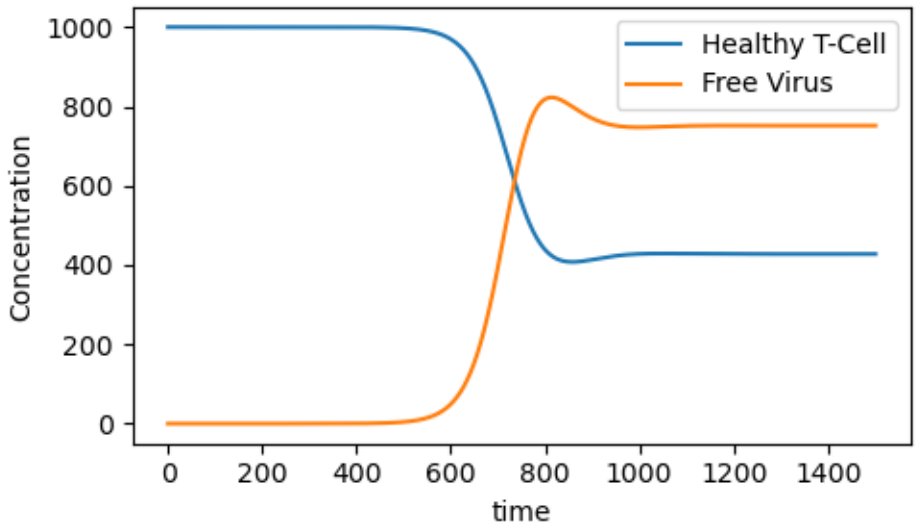
What is important to observe is the long-range or steady-state T-cell and free viral concentrations. A drop significantly below the (healthy) steady state value of 1000 cell/ml indicates the onset of AIDS.

```
def f(y, t, params):
    T, T1,T2, V = y       # unpack current values of y
    s,r,Tmax,muT,mub,muV,k1,k2,N= params  # unpack parameters
    derivs = [s+r*T*(1-(T+T1+T2)/Tmax)-muT*T-k1*T*V,    # list of dy/dt=f functions
              k1*T*V-muT*T1-k2*T1,
              k2*T1-mub*T2,
              N*mub*T2-k1*T*V-muV*V]
    return derivs

# Parameters
s=10
r=.03
Tmax=1500
muT=.02
mub=.24
muV=2.4
k1 = 2.4*10**(-5)
k2=3*10**(-3)
N=1800

# Initial values
```

(continues on next page)

(continued from previous page)

```
T0=1000
T10=0
T20=0
V0=10**(-3)

# Bundle parameters for ODE solver
params = [s,r,Tmax,muT,mub,muV,k1,k2,N]

# Bundle initial conditions for ODE solver
y0 = [T0,T10,T20,V0]

# Make time array for solution
tStop = 1501
tInc = 0.05
t = np.arange(0., tStop, tInc)

# Call the ODE solver
psoln = odeint(f, y0, t, args=(params,))

# Plot results
fig = plt.figure(1, figsize=(5,3))
plt.plot(t, psoln[:,0],label='Healthy T-Cell')
plt.plot(t, psoln[:,3],label='Free Virus')
plt.legend()
plt.xlabel('time')
plt.ylabel('Concentration')
plt.savefig("HIV1.png")
plt.tight_layout()
plt.show()
```




```
print("Steady State Healthy T-cell Concentration=",psoln[30000,0])
print("Steady State Free Viral Concentration=",psoln[30000,3])
```

```
Steady State Healthy T-cell Concentration= 427.7491644977716
Steady State Free Viral Concentration= 751.3443859716944
```

12.9.6 Sensitivity Analysis

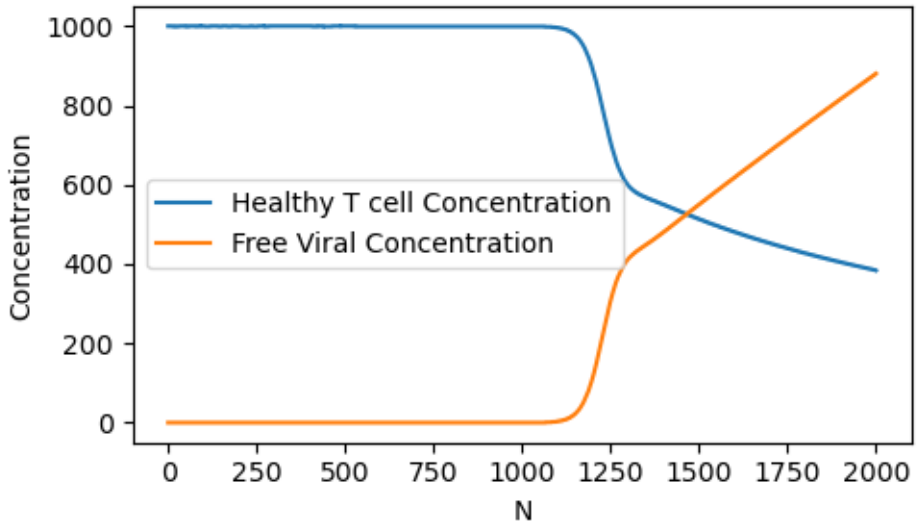
The onset of HIV is marked by rapid decrease in healthy T-cell concentration which impairs the immune system. Let us see how the long-range (i.e. steady-state) healthy T-cell concentration depends on the size of the lysing parameter N.

```
def f(y, t, params):
    T, T1,T2, V = y      # unpack current values of y
    s,r,Tmax,muT,mub,muV,k1,k2,N= params  # unpack parameters
    derivs = [s+r*T*(1-(T+T1+T2)/Tmax)-muT*T-k1*T*V,   # list of dy/dt=f functions
              k1*T*V-muT*T1-k2*T1,
              k2*T1-mub*T2,
              N*mub*T2-k1*T*V-muV*V]
    return derivs

# Parameters
s=10
r=.03
Tmax=1500
muT=.02
mub=.24
muV=2.4
k1 = 2.4*10**(-5)
k2=3*10**(-3)

# Initial values
T0=1000
T10=0
T20=0
V0=10**(-3)
# Bundle initial conditions for ODE solver
y0 = [T0,T10,T20,V0]

# Make time array for solution
tStop = 1501.
tInc = 0.05
t = np.arange(0., tStop, tInc)

# N values
Nval=np.arange(0,2001,1)
Tval=np.arange(0,2001,1)
Vval=np.arange(0,2001,1)

for N in Nval:
    # Bundle parameters for ODE solver
    params = [s,r,Tmax,muT,mub,muV,k1,k2,N]
    # Call the ODE solver
    psoln = odeint(f, y0, t, args=(params,))
    Tval[N]=psoln[30000,0]
    Vval[N]=psoln[30000,3]

fig = plt.figure(1, figsize=(5,3))
plt.plot(Nval, Tval,label='Healthy T cell Concentration')
plt.plot(Nval, Vval,label='Free Viral Concentration')
plt.legend()
plt.xlabel('N')
plt.ylabel('Concentration')
plt.savefig("Nsa.png")
plt.tight_layout()
```

(continues on next page)

(continued from previous page)

```
plt.show()
```



Note that around $N = 1200$, there is a rapid drop off in the healthy T-cell concentration. Our model predicts that an anti-retroviral drug therapy which reduces the value of N to less than 1000 would help prevent the onset of AIDS.

12.9.7 Exercises

Exercises

1) A second type of anti-retroviral drug reduces the value of k_1 by attaching themselves to free HIV viral cells so they are not able to infect healthy T-cells.

a) Using the values in Table 1 except $k_1 = 10^{-6}$ and $N = 500$, plot the healthy T cell and free viral concentration V as a function of time t over the time interval $0 \leq t \leq 1500$.

b) Repeat using $k_1 = 10^{-4}$ and $N = 500$. What do you observe?

2) Using the parameter values in Table 1 with $N = 500$, do a sensitivity analysis showing how the steady state healthy T-cell and free-viral concentrations depend on k_1.

3) Explore combinations of anti-retroviral therapies.

a) First define a function z=treatment(i, j) which assigns the values $N = 100 + i$ and $k_1 = 10^{-5}j$ and computes the corresponding steady-state healthy T-cell concentration.

b) Let $Z(i, j)$=treatment(i, j) over the range $1 \leq i \leq 100$ and $1 \leq j \leq 20$. Give the value of $Z(10, 5)$ and interpret it's meaning.

c) Make a 3D plot of your output. Label the axes appropriately.

4) What caution is needed in interpreting our model results?

5) What parts of the world have the most serious HIV/AIDS problem today?

6) A pandemic must be fought at all times by as many people as possible. What role might you play in fighting HIV/AIDS?

12.9.8 Bibliography

Africa: Using Mobile to Fight HIV/Aids in Lesotho and Kenya - Breaking the Cycle of Infection With Data, M-Money and SMS.(May 1, 2016) http://allafrica.com/stories/201605301340.html

Cohen, Myron S. et. al., 2011. Prevention of HIV-1 Infection with Early Anti-retroviral Therapy. The new England Journal of Medicine, 365(6), 493-505.

Cooper, Leon N. 1986. Theory of an immune system retrovirus. **{Proceedings of the National Academy of Science.}** 83: 9159-9163.

Covert, Douglas J. and Denise Kirschner. 2000. Revisiting Early Models of the Host-Pathogen Interactions in HIV Infection. **{Comments Theoretical Biology.}** 5(6): 383-411.

GBD 2015 HIV Collaborators. Estimates of global, regional, and national incidence, prevalence, and mortality of HIV, 1980-2015: the Global Burden of Disease Study 2015. http://www.thelancet.com/pdfs/journals/lanhiv/PIIS2352-3018(16)30087-X.pdf

Isihara, P. et. al., Immunological and Epidemiological HIV/AIDS Modeling. 2005. UMAP Journal 26.1, 49-90.

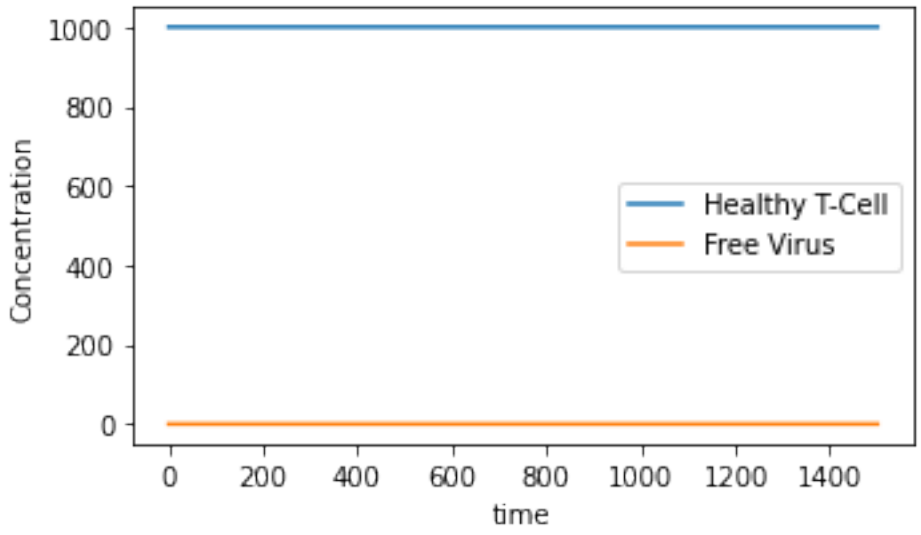
Perelson, Alan S., D.E. Kirschner and R. DeBoer. 1993. Dynamics of HIV Infection of $CD4^+$ cells. **{Mathematical Biosciences.}** 114: 81-125.

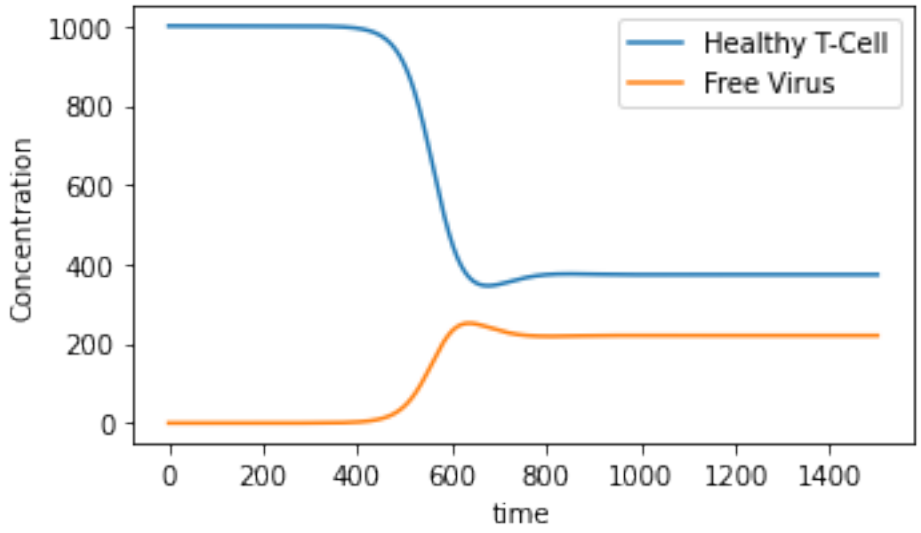
UNAIDS, 2014, 90-90-90 An ambitious treatment target to help end the AIDS epidemic. http://www.unaids.org/sites/default/files/media_asset/90-90-90_en_0.pdf

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint
```

12.10 JNB Lab Solutions

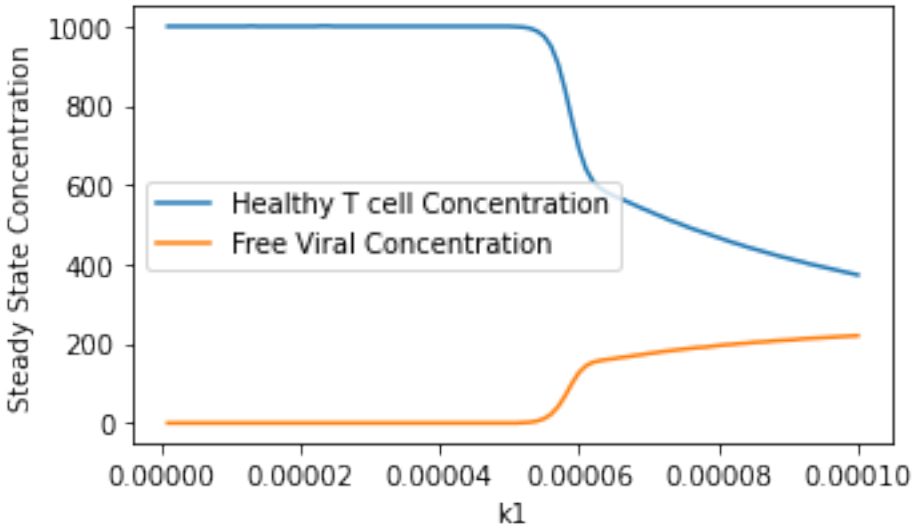
Problem 1





When $k1 = 10^{-6}$, we do not see the onset of AIDS. On the other hand, when $k1 = 10^{-4}$, the onset of AIDS occurs at around day 500.

Problem 2



Problem 3

```
# part b)
Z=np.eye(101, 21)
for i in np.arange(1,101,1):
    for j in np.arange(1,21,1):
        Z[i,j]=treatment(i,j)
Z[10,5] # steady state T-cell concentration when N=100, k1= 5 * 10**(-5)
```

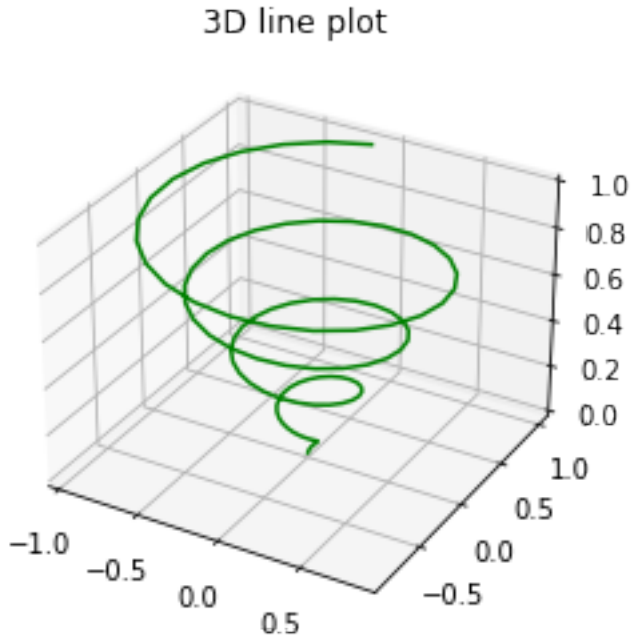
```
999.9999999998947
```

```
# importing mplot3d toolkits, numpy and matplotlib
from mpl_toolkits import mplot3d
import numpy as np
```

(continues on next page)

(continued from previous page)

```
import matplotlib.pyplot as plt

fig = plt.figure()

# syntax for 3-D projection
ax = plt.axes(projection ='3d')

# defining all 3 axis
z = np.linspace(0, 1, 100)
x = z * np.sin(25 * z)
y = z * np.cos(25 * z)

# plotting
ax.plot3D(x, y, z, 'green')
ax.set_title('3D line plot')
plt.show()
```

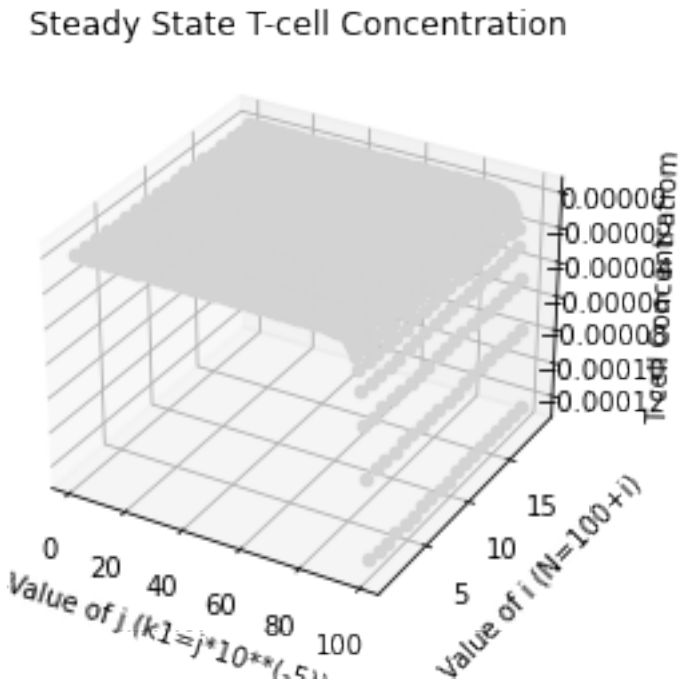



```
# part c)

# Creating dataset
x=np.arange(0,100,1)
y=np.arange(0,20,1)
z=Z

# Creating figure
fig = plt.figure()

# syntax for 3-D projection
ax = plt.axes(projection ='3d')

for i in np.arange(1,100,1):
    for j in np.arange(1,20,1):
        ax.scatter(i, j, Z[i,j],color='lightgray')
```

(continues on next page)

(continued from previous page)

```
ax.set_title('Steady State T-cell Concentration')
ax.set_xlabel('Value of j (k1=j*10**(-5))')
ax.set_ylabel('Value of i (N=100+i)')
ax.set_zlabel('T cell Concentratiom')
plt.show()
```



The graph shows the dramatic reduction of T-cell count when the values or n and/or $k1$ become sufficiently large.

CHAPTER 13

COMPLEX VARIABLES IN GROUNDWATER MODELING - Wheaton College Team

Note

This chapter applying complex variables to analyze contaminated groundwater flow was the work of a Wheaton College Team that included Dr. Paul Isihara (supervising professor), Lucy Henneker (student research assistant), and Andrew Luhmann (assisting professor), with Nathaniel Bowden, Daws Bremner, Jonathan Higgins, Stephen McKay, and Julio Reyes (Math 352 students).

This material was published previously in the ILAP Module titled A Groundwater Contaminant Extraction Model Using Complex Potentials. The UMAP Journal 43(1) (2022) 39–76. Copyright 2022 by COMAP, Inc. All rights reserved. Authosr: Paul Isihara, Lucy Henneker, and Andrew Luhmann with Nathaniel Bowden, Daws Bremner, Jonathan Higgins, Stephen McKay, and Julio Reyes. Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice. Abstracting with credit is permitted, but copyrights for components of this work owned by others than COMAP must be honored. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior permission from COMAP.

13.1 Setting the Scene

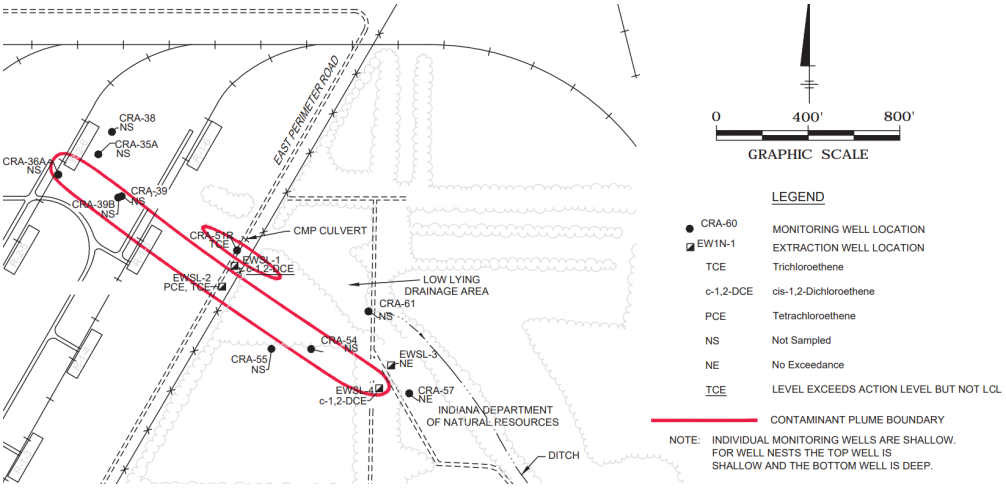
The Fisher-Calo Superfund Site is located in the multiple square-mile Kingsbury Industrial Development Park (KIDP) in La Porte County, Indiana. Two communities, Kingsbury and Kingsford Heights are located within 2 miles of KIDP. An industrial chemical processing and distribution facility operated on that site. Fisher-Calo Chemicals and Solvent Company and various subsidiaries began operations at KIDP in the early 1970s, with wastes buried on-site in thousands of drums. Spills, leakage, and a large fire in March 1978 resulted in the contamination of soil and groundwater with polychlorinated biphenyls (PCBs) and volatile organic compounds (VOCs). The contamination has spread from the source due to groundwater flow, and thus the contaminant source is not a mathematical source per se. The site's long-term remedies include:

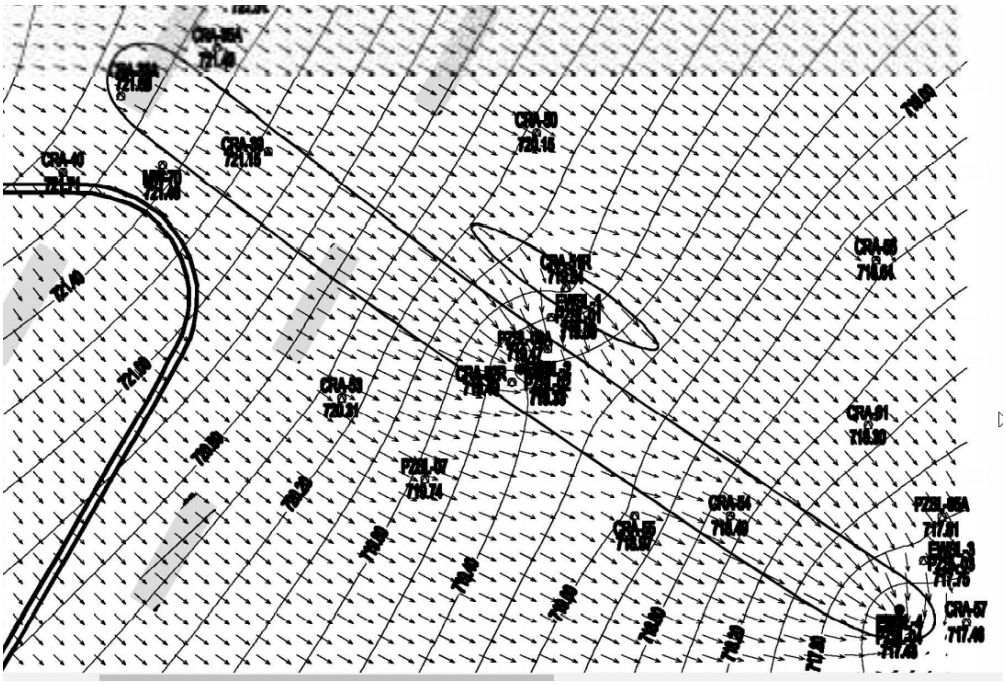
- excavation, consolidation, and offsite disposal of 3,600 drums (1994-1996);

- soil source area remediation via excavation and off site disposal of PCB impacted soils, soil vapor extraction (SVE), and bio-venting of semi-volatile organic compound (SVOC) impacted soils, and SVE of volatile organic compound (VOC) impacted soils (1996-2001); and
- groundwater extraction and treatment (1998-present) with partial Pilot Study evaluation of enhanced monitored natural attenuation (2020-present).

The EPA conducts 5 year reviews to monitor and evaluate adherence to planned operations.

Math modeling was used to support the goal to design an effective contaminant extraction well remediation system. Such a system includes both pollutant extracting wells (EW) and plume monitoring wells (MW), and is the traditional remediation approach. The Fischer-Calo Groundwater Extraction and Treatment System consisting of 16 extraction wells (EWs) began operating in Feb 1998. More than four different contaminated groundwater plumes were identified. We focus on one of these plumes, called the Space Leasing plume. In the figures below, a rough approximation of the main Fisher-Calo Space Leasing plume boundary is shown encircled in the long bold oval. For this plume, one extraction well (EWSL-2) is located mid-plume, and a second extraction well (EWSL-4) downstream at the plume's bottom right. (lower) Groundwater direction arrows are orthogonal to the lines of equipotential. Figure Credit: [U.S. Environmental Protection Agency 2020]





Groundwater at Space Leasing is present in the shallow, unconfined sand aquifer with a saturated thickness of approximately 48 to 58 feet. The direction of groundwater flow is to the southeast at an approximate horizontal hydraulic gradient of 0.002 ft/ft under 2019 pumping conditions. Groundwater in the Space-Leasing plume is impacted with chlorinated volatile organic compounds (CVOC), including trichloroethene (TCE), tetrachloroethene (PCE), and cis-1,2-Dichloroethene (cis-1,2-DCE) at concentrations greater than action levels in the upper half of the unconfined aquifer. The extent of the CVOC plume at Space-Leasing is approximately 150 feet wide by approximately 1,750 feet long and is oriented in the northwest to southeast direction.

Our model is based on 2019 data for the Space Leasing Plume remediation [U.S. Environmental Protection Agency 2020] which involved a pump and treat system with two extraction wells, one mid-plume (EWSL-2), and the other at the down-stream end (EWSL-4). We considered the following modeling problem:

Assess the containment of above action level contaminant at monitoring wells by the two extraction wells (EWSL-2, EWSL-4).

Our model utilizes the beauty and ingenuity of complex variables to analyze 2-dimensional groundwater flow. In an effort to make this Module self-contained, we begin with a brief background of complex analysis (Section 2) and its idealized application to 2-dimensional groundwater flow (Section 3). Sato's wonderful book *Complex Analysis for Practical Engineering* [Sato 2015] provides complete background for our approach. Section 4 presents our model of the Space Leasing Plume. The Supplemental Python Jupyter Notebooks (JNBs) can be used to visualize streamlines in a variety of settings, such as flow to a system of extraction wells from locations where contaminant is above action level.

13.2 Complex Analysis Background

13.2.1 The Complex Function $f(z)$

A complex function $w = f(z)$ maps a complex number $z \in \mathbb{C}$ to a complex number $w \in \mathbb{C}$ where the set of complex numbers, $\mathbb{C}$, is defined as

$$\mathbb{C} = \{x + iy : x, y \in \mathbb{R}, i^2 = -1\},$$

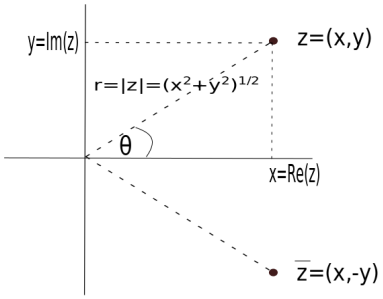
where $\mathbb{R}$ denotes the set of real numbers. Basic arithmetic operations in $\mathbb{C}$ include:

Addition and Subtraction: $z_1 \pm z_2 = (x_1 + iy_1) \pm (x_2 + iy_2) = (x_1 \pm x_2) + i(y_1 \pm y_2)$;

Multiplication: $z_1 z_2 = (x_1 + iy_1)(x_2 + iy_2) = x_1x_2 - y_1y_2 + i(x_1y_2 + x_2y_1)$; and

Division: $\frac{z_1}{z_2} = \frac{x_1+iy_1}{x_2+iy_2} = \frac{(x_1+iy_1)(x_2-iy_2)}{(x_2+iy_2)(x_2-iy_2)} = \frac{x_1x_2+y_1y_2}{x_2^2+y_2^2} + i\frac{x_2y_1-x_1y_2}{x_2^2+y_2^2}$.

Each complex number $z = x+iy$ is identified with a point $z = (x, y)$ in the Cartesian plane as shown in the figure below.



We call $x = Re(z)$ the real part of z, and $y = Im(z)$ the imaginary part. Complex numbers are also expressed in polar or exponential form as $z = r\,e^{i\theta}$, where $r = |z| = \sqrt{x^2 + y^2}$ is called the modulus of z and $\theta = \arctan\frac{y}{x}$ is called the **argument** of z (denoted $\arg z$) . Note that $|z|$ can be thought of as the ``magnitude" of z, and $|z_1 - z_2| = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ the distance between $z_1 = (x_1, y_1)$ and $z_2 = (x_2, y_2)$ as points in the Cartesian plane.

Given a complex number $z = x + iy$, the number $\overline{z} = x - iy$ is called the conjugate of z. Note that $z\,\overline{z} = x^2 + y^2 \in \mathbb{R}$. Complex conjugates are used in division since multiplication by the reciprocal of z is multiplication by $\frac{1}{z} = \frac{\bar{z}}{z\bar{z}} = \frac{x}{x^2+y^2} - i\frac{y}{x^2+y^2}$.

The output of a complex function $f : \mathbb{C} \to \mathbb{C}$ may be expressed as $f(x + iy) = u(x, y) + iv(x, y)$, where u and v are both real-valued functions. For example, if $f(z) = z^3$, then

$$f(x + iy) = (x + iy)^3 = x^3 + 3ix^2y - 3xy^2 - iy^3 = (x^3 - 3xy^2) + i(3x^2y - y^3).$$

In this case, $u(x, y) = x^3 - 3xy^2$ and $v(x, y) = 3x^2y - y^3$.

Here are some important elementary complex functions:

Polynomial functions:

$f(z) = a_0 + a_1z + a_2z^2 + ... + a_nz^n$ where $a_i \in \mathbb{C}$ for $i = 0, ..., n$ and $a_n \neq 0$. The value of the whole number n is called the degree of the polynomial. ($a_0 = 0$ is allowed for the special case $n = 0$ when $f(z)$ is constant.)

Rational functions:

$f(z) = \frac{P(z)}{Q(z)}$, where P and Q are both polynomials and $Q(z) \neq 0$.

Exponential function:

$f(z) = e^z$, where e is Euler's number ($e \approx 2.71...$). Note that $e^z = e^{x+iy} = e^xe^{iy} = e^x(\cos(y) + i\sin(y)) = e^x\cos(y) + ie^x\sin(y)$, so in this case $u(x, y) = e^x\cos(y)$ and $v(x, y) = e^x\sin(y)$. This result follows from Euler's identity

$$e^{i\theta} = \cos\theta + i\sin\theta$$

which can be proven using the Taylor expansion

$$e^z = \sum_{j=0}^{\infty} \frac{z^j}{j!}$$

that is valid for all $z \in \mathbb{C}$:

$$e^{i\theta} = \sum_{j=0}^{\infty} \frac{(i\theta)^j}{j!} = 1 + i\,\theta - \frac{\theta^2}{2!} - i\,\frac{\theta^3}{3!} + \frac{\theta^4}{4!} + i\,\frac{\theta^5}{5!} - ... =$$

$$(1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} - ...) + i\,(\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - ...) = \cos\theta + i\,\sin\theta\,.$$

The complex logarithm:

$f(z) = \log(z)$. To define the complex log function $\log z$, we use the real natural log function $\ln(\cdot)$ applied to the polar form $z = re^{i(\theta+2n\pi)}$:

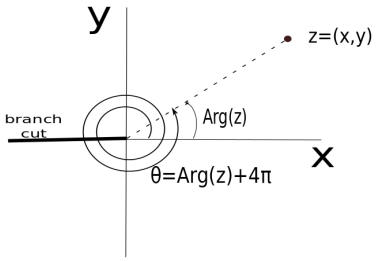
$$\log z = \ln(r) + i(\theta + 2n\pi) = \ln(|\ z\ |) + i\,\arg(z).$$

Note that since there are multiple polar representations of the same complex number z, the complex log is multiple valued as well. In order to make a single-valued function, we need to restrict the values of θ to an interval of length 2π (with one of the endpoints excluded). This restriction is called a *branch* of the natural log. A *branch cut* is a ray which determines the branch. The point from which the branch cut emanates is called the branch point. For example, the ray $\theta = -\pi$ determines a branch $-\pi < \theta \leq \pi$. This branch is called the *principal branch* and the principal value is denoted Log z (note the capital L):

$$Log\,z = \ln|z| + i\,Arg\,z,$$

where $-\pi < Arg\,z \leq \pi$.

The figure below shows the argument of z given by $\theta = \arg(z) = Arg\, z + 2n\pi$ for the case $n = 2$



Exercises

Exercises

2.1.1 Let $x = 2 + 3i$ and $y = -1 + 5i$. Calculate the following:

a) $x + y$

b) $x - y$

c) xy

d) $\frac{x}{y}$

e) $x\bar{y}$

2.1.2 Express the following complex functions in the form $u(x, y) + iv(x, y)$.

a) $f(z) = e^{2z}$

b) $g(z) = \log(z + 1)$

c) $h(z) = z^2 - 1$

2.1.3 Show that $|z_1 z_2| = |z_1||z_2|$.

2.1.4 Does $\arg(z_1 z_2) = \arg(z_1) \cdot \arg(z_2)$?

13.2.2 The Derivative $f'(z)$

The limit quotient definition of the derivative is

$$f'(z) = \lim_{h \to 0} \frac{f(z + h) - f(z)}{h}$$

(note that h is complex!)

We emphasize that in order for a complex function $f(z)$ to be differentiable at z_0, the quotient $(f(z+h)-f(z))/h$ must approach the same value along every path as $h \to 0$ including along both the real and imaginary axes.

As one might guess based on the similarity of the definitions, the basic derivative rules from real-variable calculus continue to hold in complex variables. Given that $f(z)$ and $g(z)$ are both differentiable, and c_1, c_2 are complex constants,

Linearity $(c_1 f(z) + c_2 g(z))' = c_1 f'(z) + c_2 g'(z)$

Product Rule $(f(z)g(z))' = f(z)g'(z) + g(z)f'(z)$

Quotient Rule $(\frac{f(z)}{g(z)})' = \frac{g(z)f'(z) - f(z)g'(z)}{[g(z)]^2}$

Chain Rule $[f(g(z))]' = f'(g(z))g'(z)$

Power Rule $\frac{d}{dz}z^n = nz^{n-1}$, where n is an integer,

L'Hospital's Rule $\lim_{z\to z_0} \frac{f(z)}{g(z)} = \lim_{z\to z_0} \frac{f'(z)}{g'(z)}$ for 0/0 indeterminate form limits.

Exercises

Exercises

2.2.1 Show using the limit definition that $\frac{d}{dz}(10z^3) = 30z^2$.

2.2.2 Show that the function $f(z) = \bar{z}$ is not differentiable.

2.2.3 Find $\lim_{z\to i} \frac{z^4-1}{z^8-1}$.

13.2.3 The Cauchy-Riemann Equations

The Cauchy-Riemann equations are a pair of partial differential equations that, along with conditions of continuity, are necessary and sufficient conditions for a complex function $f(z)$ to be differentiable. These equations are fundamental to both the theory and application of complex analysis.

Recall from Section 1 that the limit quotient definition of the derivative is

$$f'(z) = \lim_{h\to 0} \frac{f(z+h) - f(z)}{h}.$$

Assuming that $f(z) = f(x+iy) = u(x,y) + iv(x,y)$ is differentiable, and $h = a + ib$, we may compute $f'(z)$ along any path $(a,b) \to (0,0)$. In particular, let us choose the paths along the x and y axes:

Along the x-axis ($h = a$ where a is real):

$$f'(z) = \lim_{a\to 0} \frac{u(x+a,y) + iv(x+a,y) - [u(x,y) + iv(x,y)]}{a} =$$

$$\lim_{a\to 0} \frac{u(x+a,y) - u(x,y)}{a} + \frac{iv(x+a,y) - iv(x,y)}{a} =$$

$$\frac{\partial}{\partial x}u(x,y) + i\frac{\partial}{\partial x}v(x,y).$$

Along the y-axis ($h = bi$, where b is real):

$$f'(z) = \lim_{b\to 0} \frac{u(x,y+b) + iv(x,y+b) - [u(x,y) + iv(x,y)]}{bi} =$$

$$\lim_{b\to 0} \frac{u(x,y+b) - u(x,y)}{bi} + \frac{iv(x,y+b) - iv(x,y)}{bi} =$$

$$-i\frac{\partial}{\partial y}u(x,y) + \frac{\partial}{\partial y}v(x,y).$$

The Cauchy-Riemann equations are obtained by equating the two expressions for $f'(z)$:

$$\frac{\partial u(x,y)}{\partial x} = \frac{\partial v(x,y)}{\partial y}$$

and

$$\frac{\partial u(x,y)}{\partial y} = -\frac{\partial v(x,y)}{\partial x}$$

Note that the CR-equations are a necessary condition for differentiability: if $f(z)$ is differentiable, then the C-R equations must hold. It is also possible to prove that the CR-equations are sufficient for differentiability (see Thm 2.2 in [Sato 2015]). That is, if the C-R equations are satisfied with u_x, v_x, u_y, and v_y all continuous functions, then $f(z)$ is differentiable. In other words, assuming sufficient smoothness of $u(x, y)$ and $v(x, y)$, the CR equations are both necessary and sufficient conditions for $f(z)$ to be differentiable.

Polar Form of the Cauchy-Riemann Equations

We now consider the polar representation $f(z) = f(re^{i\theta}) = u(r, \theta) + iv(r, \theta)$ and obtain the C-R equations for u and v in polar coordinates.

Let $z = re^{i\theta}$ and let $w = z + h$. We again evaluate $f'(z)$ using two different paths $h \to 0$ (equivalently $w \to z$).

straight path

$h \to 0$ along the straight path $h = ae^{i\theta}$ (i.e. $a \to 0$ where a is real). In this case

$$f'(z) = \lim_{a\to 0} \frac{[u(r + a, \theta) + iv(r + a, \theta)] - [u(r, \theta) + iv(r, \theta)]}{ae^{i\theta}} =$$

$$\lim_{a\to 0} \frac{u(r + a, \theta) - u(r, \theta)}{ae^{i\theta}} + i\frac{v(r + a, \theta) - v(r, \theta)}{ae^{i\theta}} =$$

$$e^{-i\theta}(u_r + iv_r)$$

circular path

$w \to z$ along the circular path $w = re^{i(\theta+b)}$ ($b \to 0$ where b is real). In this case

$$f'(z) = \lim_{b\to 0} \frac{[u(r, \theta + b) + iv(r, \theta + b)] - [u(r, \theta) + iv(r, \theta)]}{re^{i(\theta+b)} - re^{i\theta}} =$$

$$\lim_{b\to 0} \frac{u(r, \theta + b) - u(r, \theta)}{re^{i\theta}(e^{ib} - 1)} + i\frac{v(r, \theta + b) - v(r, \theta)}{re^{i\theta}(e^{ib} - 1)} =$$

$$\lim_{b\to 0}[\frac{u(r, \theta + b) - u(r, \theta)}{re^{i\theta}b} + i\frac{v(r, \theta + b) - v(r, \theta)}{re^{i\theta}b}]\frac{b}{e^{ib} - 1} =$$

$$\lim_{b\to 0}[\frac{u(r, \theta + b) - u(r, \theta)}{re^{i\theta}b} + i\frac{v(r, \theta + b) - v(r, \theta)}{re^{i\theta}b}]\frac{1}{i} =$$

$$e^{-i\theta}[-i(\frac{1}{r})u_\theta + \frac{1}{r}v_\theta].$$

It follows that the C-R equations in polar form are

$$u_r = \frac{1}{r}v_\theta$$

$$\frac{1}{r}u_\theta = -v_r.$$

Exercises

Exercises

2.3.1 Use the C-R equations to show that $f(z) = z^2$ is differentiable.

2.3.2 Use the C-R equations to show that $f(z) =\mid z \mid^2$ is not differentiable.

2.3.3 Use the CR-equations to show that $f(z) = e^z$ is differentiable, and find its derivative.

2.3.4 Use the polar form of the CR-equations to find $\frac{d}{dz} \log z$.

13.2.4 Harmonic Conjugates

Harmonic functions are extremely important and have a close connection to complex differentiable functions. A real valued function $g(x, y)$ of the real variables x and y is said to be harmonic if its Laplacian is zero:

$$\nabla^2 g(x, y) = g_{xx} + g_{yy} = 0.$$

In other words, g satisfies the Laplace equation.

Let $f(z) = u(x, y) + iv(x, y)$ be a differentiable function with $u(x, y)$ and $v(x, y)$ both having continuous second partial derivatives. The C-R equation $u_x = v_y$ differentiated with respect to x gives $u_{xx} = v_{xy}$ and the other C-R equation $u_y = -v_x$ differentiated with respect to y gives $u_{yy} = -v_{yx} = -v_{xy}$, (the last equality follows from the equality of mixed partials). Adding these two results together yields

$$u_{xx} + u_{yy} = v_{xy} - v_{xy} = 0,$$

showing that u satisfies the Laplace equation.

Using similar reasoning, v must satisfy the Laplace equation as well.

Writing $f(z)$ in the polar form $f(z) = u(r, \theta) + i\, v(r, \theta)$, the polar C-R equations imply that both u and v satisfy the polar form of the Laplace equation (see Sec 2.4.2 in [Sato 2015]):

$$\frac{1}{r}\frac{\partial}{\partial r}(r\, u_r) + \frac{1}{r^2} u_{\theta\theta} = 0,$$

and,

$$\frac{1}{r}\frac{\partial}{\partial r}(r\, v_r) + \frac{1}{r^2} v_{\theta\theta} = 0.$$

This leads us now to the main topic of this section: harmonic conjugates. If the functions u and v satisfy the C-R equations for the Cartesian form or polar form, then we say v is the harmonic conjugate of u. This leads to the result that

f(z)=u+iv is differentiable if and only if v is the harmonic conjugate of u (see Thm 2.4 in [Sato 2015]).

The harmonic conjugate of u is obtainable as follows:

i) antidifferentiate $v_y = u_x$ with respect to y to obtain

$$v = U(x, y) + t(x)$$

where $t(x)$ is a real function of x only and $U_y = v_y$.

ii) differentiate the last equation with respect to x and use the C-R equation $v_x = -u_y$ to obtain

$$v_x = U_x + t'(x) = -u_y \Rightarrow t'(x) = -U_x - u_y.$$

iii) Anti-differentiating with respect to x, we obtain $t(x)$.

Example

Find the harmonic conjugate of $u(x, y) = y^3 - 3x^2y$.

Solution:

First note that u is harmonic because $u_{xx} = -6y$ and $u_{yy} = 6y$, so $u_{xx} + u_{yy} = 0$. Hence, we know that u has a harmonic conjugate, and can find it by following the three steps above:

i) $v_y = u_x = -6xy \Rightarrow v = -3xy^2 + t(x)$.
ii) $v_x = -3y^2 + t'(x) = -u_y = -3y^2 + 3x^2 \Rightarrow t'(x) = 3x^2$
iii) $t(x) = x^3 + C$, and hence $v = -3xy^2 + x^3 + C$ is the harmonic conjugate of u.

It follows that $f(z) = u + iv = (y^3 - 3x^2y) + i(-3xy^2 + x^3 + C)$ is a differentiable function.

Exercises

Exercises

2.4.1 Determine whether the following functions can be the real parts of differentiable complex functions.

a) $u_1(x, y) = x^2 - y^2$

b) $u_2(x, y) = x^2 + y^2$

c) $u_3(x, y) = x + 42$

d) $u_4(x, y) = e^x \cos y$

e) $u_5(x, y) = \sin x + \sin y$

2.4.2 For the questions to which you answered "yes" in the question above, find the harmonic conjugates.

13.2.5 Orthogonality

Another important relationship between the real and imaginary parts u and v of a differentiable function $f(z)$ is the orthogonality of their level curves. That is, the families of curves defined by $u(x, y) = p_1$ and $v(x, y) = p_2$, where p_1 and p_2 are constants, are orthogonal. We now show this to be the case. Assuming that y is a function of x, if we differentiate the level curve $u(x, y) = p_1$ with respect to x, we get

$$\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y}\frac{dy}{dx} = 0,$$

so the slope along this level curve is computable using

$$\frac{dy}{dx} = -\frac{\partial u}{\partial x} \Big/ \frac{\partial u}{\partial y}$$

Exercises

Exercises

2.5.1 Let $\Phi(x, y) = -x$. Determine $\Psi(x, y)$ such that $\Omega(z) = \Omega(x + iy) = \Phi(x, y) + i\Psi(x, y)$ is differentiable and explain why $\Phi(x, y) = c_1$ and $\Psi(x, y) = c_2$ defines an orthogonal family of curves.

2.5.2 Let $f(z) = z^2$.

a)Find $u(x, y)$ and $v(x, y)$ such that $f(z) = u(x, y) + iv(x, y)$.

b) Show that the family of curves $u(x, y) = c_1$, $v(x, y) = c_2$ are orthogonal.

13.3 Idealized Groundwater Flow

13.3.1 Two-Dimensional Flow and the Continuity Equation

Having introduced differentiable (also called analytic) functions in Section 2, we now turn to the topic of how these functions are used to model idealized 2-dimensional groundwater flow. We will also show how isolated singularities (points where a function is not differentiable) are used to describe sources or sinks. In modeling groundwater flow, we assume that the flow is two dimensional with x corresponding to the east-west and y to the north-south directions. In particular, such flow in 3 dimensions does not depend on z; that is, any plane parallel to the (x, y)-plane has an identical flow pattern. The groundwater flow equation used to model such flow in 3 dimensions results when the *continuity equation* is combined with Darcy's Law (Section 3.2) The continuity equation relates the rate of change in density at a given point (x, y, z) to the divergence of the mass flow rate. This flow rate is the product of the density $\rho(x, y, z; t)$ (mass per unit volume at the position (x, y, z) and time t) times the fluid velocity $\vec{v}(x, y, z; t) = \mathbf{V}$ at the same point and time. The velocity is assumed to be continuously differentiable.

The continuity equation is given by the equation

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{V} = 0. \tag{13.1}$$

This equation is essentially a statement of the conservation of mass, meaning mass is neither created nor destroyed. Note that the continuity equation does not hold at a source or sink since the velocity is undefined at such points.

The continuity equation may be understood mathematically by defining a "control" volume region $\mathcal{V}$ bounded by a surface $\mathcal{S}$. Fluid flows into and out of this control volume. Since no mass is created or destroyed within $\mathcal{V}$, any increase in the fluid density within $\mathcal{V}$ over time is due solely to mass flowing into $\mathcal{V}$ through the boundary surface $\mathcal{S}$. We express this relationship as

$$\iiint_{\mathcal{V}} \frac{\partial \rho}{\partial t}\, dV = -\iint_{\mathcal{S}} \rho \mathbf{V} \cdot \hat{\mathbf{n}}\, dS,$$

where dV is an infinitesimal volume element, dS is an infinitesimal surface area element, and $\hat{\mathbf{n}}$ is the outward unit vector normal to the surface. Using the divergence theorem, we have:

$$\iiint_{\mathcal{V}} \frac{\partial \rho}{\partial t}\, dV + \iiint_{\mathcal{V}} (\nabla \cdot \rho \mathbf{V})\, dV = \iiint_{\mathcal{V}} \frac{\partial \rho}{\partial t} + (\nabla \cdot \rho \mathbf{V})\, dV = 0.$$

Finally, since the mathematical volume $\mathcal{V}$ is arbitrarily chosen and could be infinitesimal, the integrand in the last integral must itself be equal to 0:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{V} = 0,$$

which is the continuity equation previously stated. For an *incompressible* fluid, $\frac{\partial \rho}{\partial t} = 0$, which is also an underlying assumption about the fluid flow discussed in this module.

Exercise

Exercises

3.1.1 For the density function $\rho(x, y) = 2e^{-y}$ and the velocity vector $\mathbf{V}(x, y) = \langle x^2 + y^2, 2x \rangle$, compute the following to evaluate whether $\rho(x, y)$ and $\mathbf{V}(x, y)$ satisfy the continuity equation:

a) $\frac{\partial \rho}{\partial t}$

b) $\nabla \cdot \mathbf{V}$

c) $\frac{\partial \rho}{\partial t} + \nabla \cdot \rho\mathbf{V}$. Does this equal 0 as in the continuity equation?

13.3.2 Darcy's Law and the Velocity Potential Φ

To analyze groundwater flow, we note that the velocity vector $\mathbf{V}$ depends on the type of liquid, the characteristics of the surrounding material, and the pressure at any given point. Darcy's Law summarizes these relationships. For the groundwater flows we are considering, Darcy's Law states that the velocity $\mathbf{V}$ is given by

$$\mathbf{V} = -\frac{k}{\mu}\nabla p,$$

where k is the permeability of the surrounding material, μ is the viscosity of the liquid, and p is the pressure. The fluid flow moves in the direction from higher to lower pressure, that is, the direction of $-\nabla p$. Incidentally, Darcy's Law has many analogues in other physical situations, such as heat transfer or electricity and magnetism. All of these feature vector fields which depend on the negative gradient of a scalar quantity.

From vector calculus, we are familiar with conservative vector fields that can be written as the gradient of a scalar potential function. In the case of Darcy's Law, the fluid velocity $\mathbf{V}$ has a *velocity potential* $\Phi = \frac{k}{\mu}p$ such that

$$\mathbf{V} = -\nabla\Phi.$$

Note that fluid flows from higher to lower values of the velocity potential Φ since we remarked earlier that fluid flows from higher to lower pressure p.

Since we assume that the groundwater flow is incompressible (constant density ρ), the continuity equation reduces to

$$\nabla \cdot \mathbf{V} = 0.$$

Combining these results, we see that for a 2-dimensional flow, the potential $\Phi(x, y)$ satisfies Laplace's equation:

$$\nabla \cdot \mathbf{V} = -\nabla \cdot (\nabla\Phi) = 0 \Rightarrow \nabla \cdot (\Phi_x \vec{i} + \Phi_y \vec{j}) = 0 \Rightarrow \Phi_{xx} + \Phi_{yy} = 0.$$

In other words, for the flows we consider, the velocity potential $\Phi(x, y)$ will be a harmonic function.

Exercises

Exercises

3.2.1 For the density function $\rho(x, y) = 2x^2e^{-y}$, compute $\nabla\rho$.

3.2.2 Given the velocity vector $\mathbf{V}(x, y) = \langle 2xy, x^2 - y^2 \rangle$,

a) Show that $\nabla \cdot \mathbf{V} = 0$.

b) Find a function Φ such that $\mathbf{V} = -\nabla\Phi$.

c) Show that the Φ you obtained in part b) is harmonic.

13.3.3 The Stream Function Ψ and Complex Potential Ω

Since all harmonic functions have a harmonic conjugate, we know that the 2-dimensional velocity potential $\Phi(x, y)$ from the previous section has a harmonic conjugate which we designate $\Psi(x, y)$ and call a *stream function*.

$\Phi(x, y)$ and $\Psi(x, y)$ are related by the Cauchy-Riemann (C-R) Equations.

Cartesian form:

$$\frac{\partial\Phi}{\partial x} = \frac{\partial\Psi}{\partial y}$$

$$\frac{\partial\Phi}{\partial y} = -\frac{\partial\Psi}{\partial x},$$

Polar form:

$$\frac{\partial\Phi}{\partial r} = \frac{1}{r}\frac{\partial\Psi}{\partial\theta}$$

$$\frac{1}{r}\frac{\partial\Phi}{\partial\theta} = -\frac{\partial\Psi}{\partial r}.$$

Given a velocity potential $\Phi(x, y)$, we can apply our method from the section on Harmonic Conjugates to find a stream function $\Psi(x, y)$, and thereby obtain a differentiable function $\Omega(z) = \Phi + i\Psi$ called a *complex potential*. The level curves Φ=constant are called *equipotentials*, and the level curves Ψ = constant are called *streamlines*. Our 2D groundwater modeling approach is based on the complex potential Ω, and its associated velocity potential $\Phi = Re(\Omega)$ and stream function $\Psi = Im(\Omega)$.

Example

Given the velocity potential $\Phi(r) = k \ln r$ where =$\sqrt{x^2 + y^2}$ and k is a real constant, find the stream function Ψ. Show that the equipotentials and streamlines are orthogonal.

Solution.

First we take the partial derivative of Φ with respect to r,

$$\frac{\partial}{\partial r}\Phi = \frac{\partial}{\partial r}k \ln r = \frac{k}{r}\,.$$

Using the polar Cauchy-Riemann (C-R) equation $\frac{\partial\Phi}{\partial r} = \frac{1}{r}\frac{\partial\Psi}{\partial\theta}$, we have

$$\frac{\partial\Phi}{\partial r} = \frac{k}{r} = \frac{1}{r}\frac{\partial\Psi}{\partial\theta} \Rightarrow \frac{\partial\Psi}{\partial\theta} = k \Rightarrow \Psi = k\theta + t(r)$$

Next we take the derivative of Ψ with respect to r and use the other polar C-R equation $-\frac{\partial\Psi}{\partial r} = \frac{1}{r}\frac{\partial\Phi}{\partial\theta}$ to solve for $t(r)$,

$$-t'(r) = \frac{1}{r}\frac{\partial\Phi}{\partial\theta} = 0 \Rightarrow t(r) = \ constant.$$

Thus, the stream function is $\Psi = k\theta + \ constant$. If we let $t(r) = 0$, then $\Omega = \Phi + i\Psi = k \ln r + ik\theta = k \log(z)$. For this complex potential $\Omega(z) = k \log z$, the streamlines θ = constant are radial lines which include the origin.

Note that the equipotentials are given by $\ln r$ = constant or equivalently r = constant and hence are concentric circles centered at the origin. This illustrates the general fact that the streamlines and equipotentials are orthogonal.

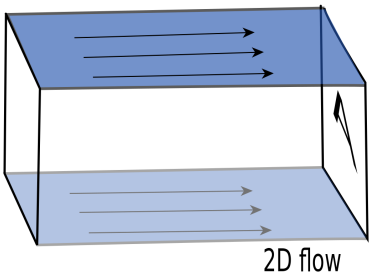
Exercise

Exercise

3.3.1 Let $\Phi(x) = -\frac{q_u}{A}x + \Phi_0$ be the velocity potential for uniform flow. Find the corresponding stream function Ψ and complex potential Ω. Then show that the equipotentials and streamlines are orthogonal.

13.3.4 Uniform Flow

Uniform flow means both the flow rate q_u ($\frac{m^3}{s}$) and the velocity vector $< V_x, V_y >$ are constant at each point. If the flow is in the x direction, then $V_y = 0$ and $V_x = q_u/A$, where A is the cross-sectional area through which such flow in 3 dimensions passes (see figure below). The flow pattern is exactly the same in every plane parallel to the original.



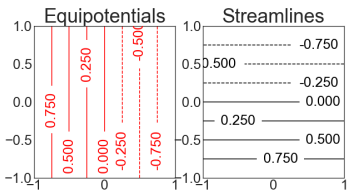
In this case, Darcy's Law becomes

$$V_x = \frac{q_u}{A} = -\frac{d\Phi}{dx}.$$

Integrating, we get

$$\Phi(x) = -\frac{q_u}{A}x + \Phi_0.$$

For example, set $-\frac{q_u}{A} = 1$ and $\Phi_0 = 0$ so in this case $\Phi = -x$. Since $\Psi_y(x, y) = \Phi_x = -1$, we can let $\Psi(x, y) = -y$ and obtain the complex potential is $\Omega(x, y) = -x - iy = -z$



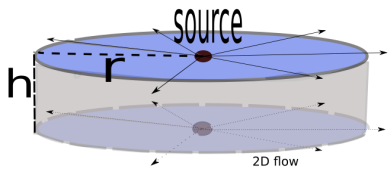
Exercise

Exercise

3.4.1 Consider the uniform flow with complex potential $\Omega(z) = z$. Find Φ and Ψ, and describe the direction of the flow.

13.3.5 Sources and Sinks

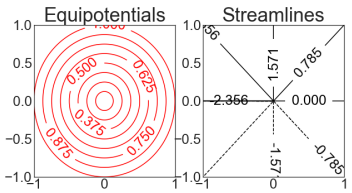
For a source or sink with a constant flow rate q_w ($\frac{m^3}{s}$) in the r (radial) direction ($q_w > 0$ for a source and $q_w < 0$ for a sink), the area A through which such flow in 3 dimensions passes is the lateral surface of a circular cylinder with radius r and height h: $A = 2\pi rh$ (see figure below). The flow pattern is assumed to be exactly the same in each plane parallel to the original.



Darcy's law becomes

$$V_r = \frac{q_w}{2\pi rh} = -\frac{d\Phi}{dr},$$

so $\Phi(r) = -\frac{q_w}{2\pi h} \ln r$. Since fluid flows from higher potential to lower potential, the velocity potential Φ decreases with increasing r for a source ($q_w > 0$) and increases with increasing r for a sink ($q_w < 0$). The equipotential lines (level curves for Φ) have the form $\ln(r) = constant$, or more simply r = constant. The equipotentials are concentric circles centered at the origin.



Setting $\Phi_0 = 0$, as we showed earlier in introducing the stream function and complex potential, a corresponding stream function is $\Psi = -\frac{q_w}{2\pi h}\theta = -\frac{q_w}{2\pi h} \arg(z)$ so the streamlines have the form $\Psi = -\frac{q_w}{2\pi h} \arg z = -\frac{q_w}{2\pi h} \arctan(\frac{y}{x})$ = constant, or more simply, θ = constant. The streamlines are half-lines which radiate to or away from the origin. Note that in this case, we can use a potential of the form $\Omega(z) = -\frac{q_w}{2\pi h}(\ln r + \theta) = -\frac{q_w}{2\pi h} \log(z)$ to model a sink ($q_w < 0$) or a source ($q_w > 0$) at the origin. (More generally, if the source or sink is located at $z_w = (x_w, y_w)$, then a complex potential of the form $\Omega(z) = -\frac{q_w}{2\pi h} \log(z - z_w)$ is used to model the source or sink.)

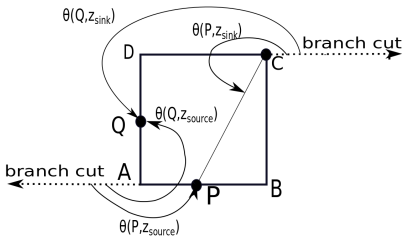
Example

Consider a complex potential for an equal strength source ($q_w/h = 1$) at A(0,0) and sink ($q_w/h = -1$) at C(1,1): $\Omega(z) = \frac{-1}{2\pi}[\log(z) - \log(z - (1 + i))]$. In this case, the velocity potential is given by

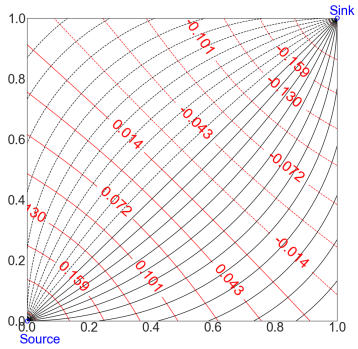
$$\Phi(x, y) = \frac{-1}{2\pi}[\sqrt{x^2 + y^2} - \sqrt{(x - 1)^2 + (y - 1)^2}].$$

The stream function must account for the choice of branch cuts. Suppose we specify the branches as shown in the figure below, namely a branch cut in the direction of the negative x-axis at the branch point A(0,0) and a branch cut in the direction of the positive x-axis at the branch point C(1,1). The stream function is in this case

$\Psi(x, y) = \frac{-1}{2\pi}[\theta(z, z_{source}) - \theta(z, z_{sink})]$, $z = x + iy$.



The equipotentials and streamlines are shown in the figure below.



13.3.6 The Rankine Oval

The superposition principle for flows (based on superposition of harmonic functions) says that to find the complex potential $\Omega(z)$ for a uniform flow with a source at d_1 and a sink at d_2, we add together the three individual complex potentials:

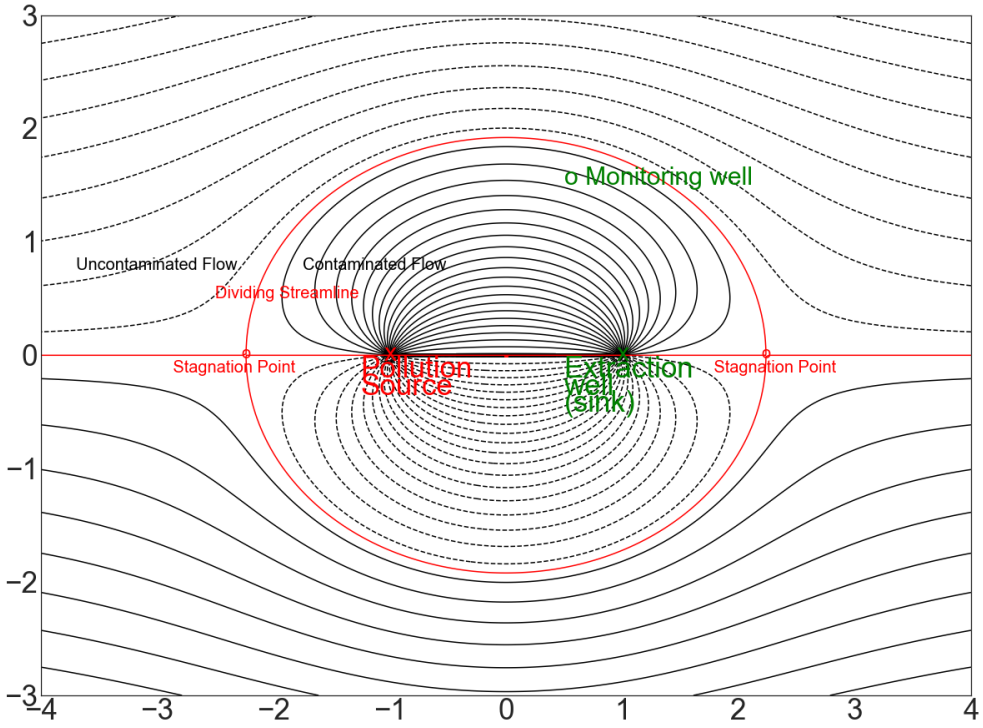
$$\Omega(z) = -\frac{q_u}{A}z - \frac{q_{source}}{2\pi h}\log(z-d_1) - \frac{q_{sink}}{2\pi h}\log(z-d_2).$$

If the flow rates are balanced (i.e. have equal strength $q_{source} = -q_{sink} = q_s$), and the sink and source positioned symmetrically about the origin ($-d_1 = d_2 = d > 0$) then the complex potential has the form

$$\Omega(z) = -\frac{q_u}{A}z - \frac{q_s}{2\pi h}\log(\frac{z+d}{z-d}),$$

where $2d$ is the distance between the source and sink.

This complex potential $\Omega(z)$ can be used to model uniform flow around an oval boundary known as Rankine's oval which contains in its interior the flow from source to sink. The figure below shows streamlines for the complex potential $\Omega(z) = -z - 2\log(\frac{z+1}{z-1})$. A source is located at (-1,0) and equal strength sink at (1,0). Rankine's oval is the dividing streamline between uncontaminated and contaminated flow. A separation of the flow occurs at the stagnation point where the oval intersects the horizontal axis through the center of the oval.



In general, the complex velocity W is defined to be the negative of the derivative of Ω. That is, $W = -\frac{d\Omega}{dz}$. The stagnation points, or critical points where the velocity is zero and thus the flow can abruptly change directions, are obtained by setting $W = 0$. In our example,

$$W = -\frac{d\Omega}{dz} = \frac{q_u}{A} + \frac{q_s}{2\pi h}\left(\frac{1}{z+d} - \frac{1}{z-d}\right) = 0\,,$$

which yields the stagnation points $z = \pm\sqrt{d^2 + \frac{A}{q_u}\frac{q_s}{\pi h}d}$. A streamline which passes through a stagnation point is called a **dividing streamline**. The value of the stream function Ψ at a stagnation point gives the value of the level curve representing the dividing streamline. If we think of the source as a pollution source, then the Rankine oval is the dividing streamline passing through the two stagnation points and thus indicates the boundary between uncontaminated and contaminated flow.

In the limit as $d \to 0$ and $m = \frac{q_s}{2\pi h} \to \infty$ in such a way that $2dm = \sigma$ remains finite, a dipole is created with a circular Rankine oval. In this case, the complex potential takes the form $\Omega(z) = -\frac{q_u}{A}z - \frac{\sigma}{z}$ (see Exercise 3) and is used to represent uniform flow around a circular boundary.

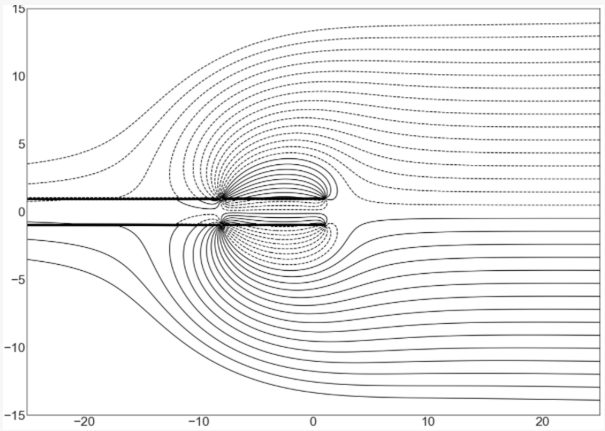
Exercises

Exercises

3.6.1 What scenario is modelled by the complex potential $\Omega = -z - 2\log(z+1)$? Make a sketch of the streamlines corresponding to this potential.

3.6.2 The figure below shows the streamlines for the potential

$$\Omega(z) = -z - 4\log(z-(-8+i)) + 2\log(z-(1+i)) - 4\log(z-(-8-i)) + 2\log(z-(1-i))$$



a) Find the stream function $\Psi(x, y)$.

b) How many sinks and sources are there?

c) Which have greater strength, the sources or the sinks?

d) If contaminant flows from the sources along the streamlines, will it be contained by the sinks (extraction wells)?

e) What are the approximate coordinates of the stagnation points?

f) What equation is used to find the exact coordinates of the stagnation points?

3) Use a centered-difference quotient definition of derivative $\frac{d}{dz}\log z = \frac{\log(z+d)-\log(z-d)}{2d}$ to show that

$$-\lim_{d\to 0}\frac{q_s}{2\pi h}\log\frac{z+d}{z-d} = -\frac{\sigma}{z}$$

where $m = q_s/(2\pi h)$ and $\sigma = 2dm$.

13.4 Contaminant Extraction Modeling

Though the idea of streamline modeling of 2-dimensional groundwater flow using a complex potential $\Omega(z)$ is conceptually elegant, in reality, groundwater flow is more complicated to analyze. Being by nature 3 dimensional, a polluted groundwater flow (plume) is normally analyzed using field data that is processed using sophisticated software such as MODFLOW. Good data and advanced software alone, however, will not guarantee that a 3 dimensional model gives good results. As has unfortunately been the case in real-life, errors in assumptions used to design a pollutant well extraction ("pump-and-treat") system can result in a counter-productive remediation that wastes many hours of labor and millions of dollars. Essential features of a good model include

- good conceptual model (see the first section called Setting the Scene);

- good assumptions on boundary conditions, initial conditions, and hydraulic parameters;
- appropriate calibration;
- consideration of lateral (horizontal) properties; and
- analysis of multiple (vertical) layers.

An aquifer which is sufficiently isotropic (permeability is constant with direction) and homogeneous (permeability is constant with location) may be amenable to a relatively simple 2-dimensional model. Data from the Space Leasing plume at the Fischer-Calo Superfund site involving monitoring and extraction wells at different depths and locations (see the table below) suggest an aquifer flow system that behaves in a fairly ideal fashion, leading us to develop a 2-dimensional model to simulate flow through this porous media (sand).

Level	Well Name	x	y	Φ	k
NE	CRA-35A	−10.5	2.3	721.49	0.11
XX	CRA-36A	−11.8	0	721.69	0.11
NE	CRA-38	−10.6	3.8	721.51	0.11
XX	CRA-39	−8.1	0.9	721.15	0.12
–	CRA-40	−11.8	−2.3	721.71	0.11
–	CRA-50	−2.3	4.8	720.15	0.14
XX	CRA-51R	−0.2	2.1	719.84	0.20
–	CRA-52R	0	−0.5	719.49	0.46
–	CRA-53	−3.2	−3.2	720.31	0.14
XX	CRA-54	6.2	−0.3	718.49	0.14
NE	CRA-55	4.2	−1.6	718.97	0.15
–	CRA-56	5.2	6.8	718.64	0.11
NE	CRA-57	12.4	0.6	717.46	0.13
–	CRA-61	7.1	3.3	718.30	0.12
XX	EWSL-1	0	1.25	–	–
XX	EWSL-2	0	0	–	–
NE	EWSL-3	10.8	2	–	–
XX	EWSL-4	10.8	0	–	–
NE	MW-67	−14.9	−2.2	722.11	0.10
–	MW-69	−11	6.7	721.57	0.11
–	MW-70	−10.1	−0.9	721.46	0.11
–	PZSL-05A	10.4	2.5	717.81	0.13
–	PZSL-06	11.5	−2.3	717.60	0.12
–	PZSL-07	−0.5	−3.7	719.74	0.15
–	PZSL-08A	0.2	0.5	719.47	0.43

This table summarizes the data used in analysis of the basic model's system well extraction strength parameter k. Level designations: NE=did not exceed action level Volatile Organic Compound (VOC) contaminant concentration in past 5 years; XX=above action level contaminant. A dash (-) for Level indicates data that was not sampled/available. The well name codes CRA , MW, and PZSL refer to monitoring wells, and EWSL to extraction wells (only positions and levels were considered for the latter). (x, y) coordinates are the approximations used in a reduced-scale horizontal plume model. The extraction well strength k values has an average strength $k = k_{ave} \approx .16$. Well levels, names, locations, and potentials are based on EPA Fifth 5-Year Review Space Leasing Plume data included in [U.S. Environmental Protection Agency 2020].
Summary of data used in analysis of the basic model's system well extraction strength parameter k. Level designations: NE=did not exceed action level Volatile Organic Compound (VOC) contaminant concentration in past 5 years; XX=above action level contaminant. A dash (-) for Level indicates data that was not sampled/available. The well name codes CRA , MW, and PZSL refer to monitoring wells, and EWSL to extraction wells (only positions and levels were considered for the latter). (x, y) coordinates are the approximations used in a reduced-scale horizontal plume model. The extraction well strength k values has an average strength $k = k_{ave} \approx .16$. Well levels, names, locations, and potentials are based on EPA Fifth 5-Year Review Space Leasing Plume data included in [U.S. Environmental Protection Agency 2020].

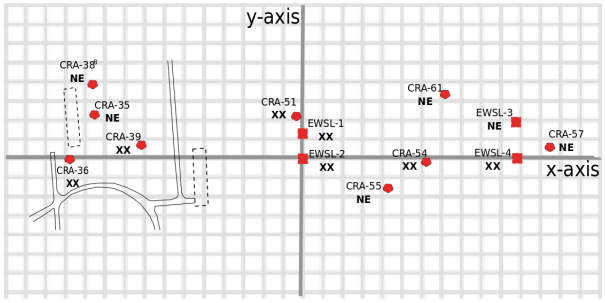
13.4.1 Model Assumptions

Basic model assumptions include

1) The Space Leasing plume is sufficiently isotropic and homogeneous and flow is generally horizontal to justify a relatively simple 2D analysis. In particular, we assume that above action level contaminant may depend on position but not on depth.

2) The contaminant behaves conservatively and moves along groundwater streamlines. Thus, our model represents convection, but excludes natural processes that include diffusion, dispersion, and chemical reaction.

3) A reduced-scale model of the plume may be positioned horizontally as in Figure 12 (shown below). The extraction wells (denoted EW) are thereby located at $z_{EWSL-2} = (0, 0)$ and $z_{EWSL-4} = (10.8, 0)$, and uniform flow is from left to right



This figure shows the space leasing plume positioned horizontally with midplume extraction well EWSL-2 at the origin and downstream extraction well EWSL-4 at (10.8,0). Wells marked NE did not reach action level Volatile Organic Compound (VOC) contaminant concentrations in the past 5 years. Those marked XX exceeded action levels and are the targeted contaminant locations in this study. (Image based on data in [U.S. Environmental Protection Agency 2020]

4) Contaminant which is above action level must be contained by the two extraction wells. (Above action level contaminant that is not contained by the two well system is considered to be part of a secondary plume requiring an alternate method of remediation.) Hence, our goal for an effective system is that all streamlines from above action level monitoring locations to be a subset of the streamlines which end at the two extraction wells EWSL-2 (mid-plume) and EWSL-4 (downstream).

5) Based on Oct. 2019 data, the EWSL-2 (mid-plume) has a flow-rate of 24.6 gallons/minute (GPM), and EWSL-4 (downstream) is 16.1 GPM; we assume the strengths of the extraction wells are in a constant 3:2 ratio even though the extent of the plume today has certainly been impacted by a more complex history of pumping.

13.4.2 Methods

We model the pollution extraction system using a complex potential of the form

$$\Omega(z) = k_0 z + k_1 \log(z - z_{EWSL-2}) + k_2 \log(z - z_{EWSL-4}) + C,$$

where $C = \Phi_0$ is a real constant. This is based on our discussions of modeling uniform flow and sinks. As we did in the Rankine oval problem, we use the superposition principle to obtain the complex potential $\Omega(z)$ by adding together the complex potentials for uniform flow and the two extraction wells modeled as sinks. We determine the coefficients as follows:

1) based on an equipotential map for the SL plume [U.S. Environmental Protection Agency 2020], we set the baseline potential at $\Phi_0 = 717$, so $C = 717$.

2) Since the strengths of the mid-plume and downstream extraction wells are in a 3:2 ratio, we let $k_1 = 3k$ and $k_2 = 2k$. We call k the system's *extraction well strength parameter*. It follows that the velocity potential has the form

$$\Phi(z) = \Phi(x + iy) = k_0 x + 3k \ln(|\, z - z_{EWSL-2}\,|) + 2k \ln(|\, z - z_{EWSL-4}\,|) + 717.$$

3) Under the assumption of constant horizontal uniform flow, if we estimate k_0 using two equipotentials, one at $x = x_1$ with value Φ_1 and one at $x_1 + d$ with value Φ_2, then $k_0 x_1 = \Phi_1$ and $k_0(x_1 + d) = \Phi_2 \Rightarrow k_0 \approx \frac{\Phi_2 - \Phi_1}{d}$. Using this process on three pairs of upstream equipotentials in a horizontally rotated version of Figure 12 (lower) and then averaging, we obtained the estimate $k_0 \approx -.145$.

4) We study the system's extraction well strength-parameter k by two related methods:

A) **System Sensitivity Method**

Using the values of Φ at the monitoring wells in the above Table and computing the strength parameter k gives a range of values for k between roughly .1 and .5 with an average value $k_{ave} \approx .16$. In order to better understand the significance of $k_{ave} \approx .16$, we explore the types of contaminant extraction behavior that can be observed for a wide range of k values.

B) **Critical Value Method**

Since we assume that an effective system must capture all targeted above action level pollutant, we look for a critical value $k = k^*$ at which all monitoring wells with above action level contaminant are contained within the dividing streamlines for the extraction wells. The critical value $k = k^*$ is the smallest value of the extraction well strength parameter for which the well system is effective. A second critical value $k = \hat{k}$ indicates the strength parameter level above which the system becomes inefficient. In this case, we take inefficient to mean that the mid-plume extraction well EWSL-2 can be turned off and the targeted contaminant captured using only the downstream extraction well EWSL-4 without increasing the latter well's strength.

13.4.3 Model Output

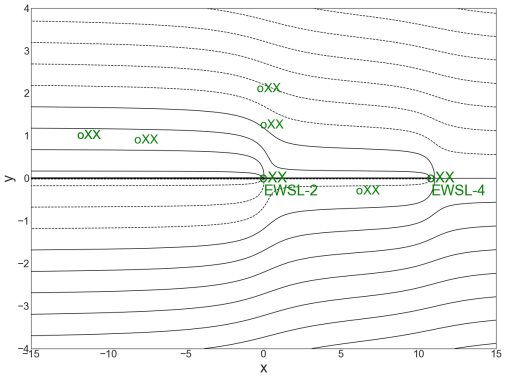
A) System Sensitivity Method

The System Sensitivity Method explores a wide range of k values to arrive at a system classification.

1) **Ineffective System** (k is too low to capture the targeted contaminant.)

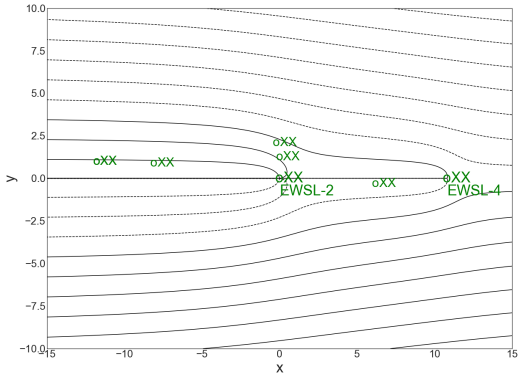
An ineffective system occurs when $k = k_{ave} \approx .16$ as the system fails to capture streamlines from two of the targeted monitoring well locations with above action level contaminant.

$k = k_{ave} \approx .16$



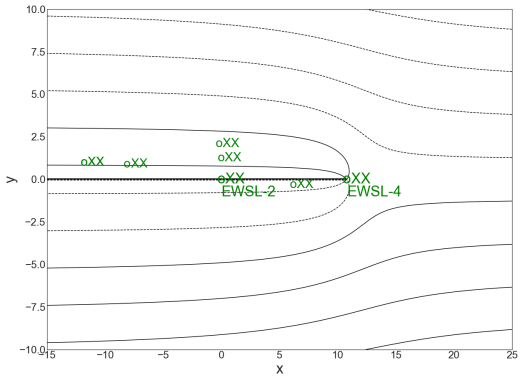
2) **Regular System** (k is effective and efficient.)

A regular system such as occurs when $k = .4$ is characterized as one which must use both the mid-plume well (EWSL-2) and the downstream extraction well EWSL-4 to capture streamlines emanating from all the targeted above action level positions.



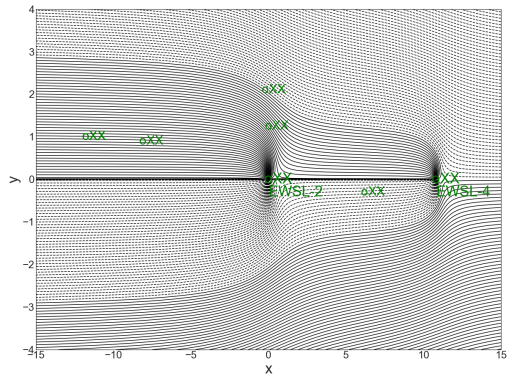
3) **Inefficient System** (k is sufficiently high that the midplume well could be turned off and still maintain system effectiveness.)

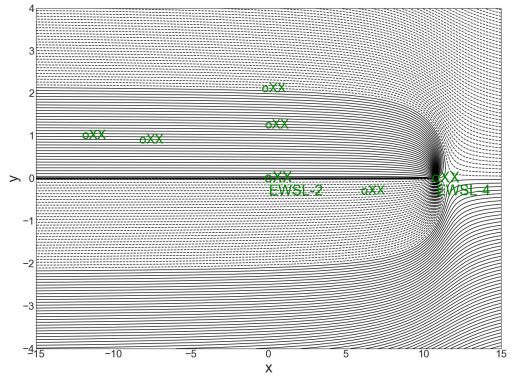
An inefficient system such as occurs when $k = 1$ is such that the downstream well (EWSL-4) operating by itself could capture all the streamlines from the targeted above action level positions, thus rendering the midstream well EWSL-2 unnecessary. The figure below shows that all the targeted streamlines are captured by EWSL-4 when the strength of EWSL-2 has been reduced to 0.



B) Critical Value Method

The critical value method focuses on "critical" values between effective/ineffective systems and between regular/inefficient systems (such critical values may be discovered using the System Sensitivity Method). As shown in the figures below, we found the critical value $k = k^* \approx .285$ is the smallest k for which the two well system effectively captures streamlines from all the targeted action level positions (XX). The system is ineffective for $k < k^*$. A second critical value $k = \widehat{k} \approx .51$ marks the highest value of k for which both wells are needed to capture all the targeted contaminant. The system is inefficient when $k > \widehat{k}$. A regular system is one whose k value satisfies $k^* < k < \widehat{k}$. That is, the system is both effective and efficient.





13.4.4 Sensitivity Analysis

The System Sensitivity Method is used to classify systems as being ineffective, regular, or inefficient. The Critical Value Method identifies numerically the critical values $k = k^* \approx .285$ dividing ineffective and regular systems and $k = \hat{k} \approx .51$ dividing regular and inefficient systems. Our estimate that $k \approx k_{ave} = .16$ is in the range of ineffective systems.

We can repeat the analysis only reversing the relative pumping strengths of EWSL2:EWSL4 from 3:2 to 2:3. Such a change would presumably maintain the same system cost and carbon footprint. As shown in the table below, both critical values k^* and $\hat{k}$ decrease, indicating that reduced pumping strength is required for the 2:3 system to be regular (both effective and efficient). However, for $k = k_{ave} \approx .16$, this reversal in pumping strength ratios would not change the ineffectiveness of the system.

Ratio EWS-2:EWS-4	k^*	$\hat{k}$
3:2	0.28	0.51
2:3	0.25	0.34

Intuitively, when k is small, the total volume of pumped out water remains roughly the same for both pumping strength ratios since the midplume well does not extract much water from downstream. Therefore, the total extent of the combined capture zones for the two extraction wells remains roughly the same. On the other hand, for higher k values where the mid-stream well EWSL-2 extracts more downstream water (contributing to inefficiency), the overall system performance

will improve if the strength ratio is reversed from 3:2 to 2:3. (Note: Removing three outliers in the data for potentials, our model estimates $k_{ave} \approx .12$. This suggests that an effective system might be achievable by reversing the strength ratio to 2:3 and roughly doubling the pumping rates. Additional data would be needed to check the correlation between the parameter k and the pumping rates for the range of k values under consideration.)

13.4.5 Discussion

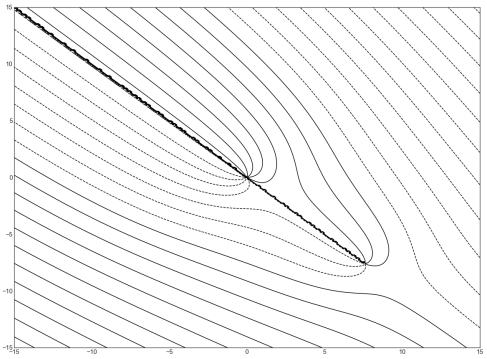
The 2D groundwater flow modeling we have introduced in this Module is an elegant application of complex variables. Moreover, complex potentials are useful in many other physical contexts such as heat conduction, and also gravitational and electrostatic fields. An important strength of using a complex potential $\Omega(z)$ for groundwater modeling is that it gives a clearer picture of how contaminated flow can be captured by extraction wells. Direction arrows which are perpendicular to equipotential lines give the direction of streamlines but not a clear indication of dividing streamlines. Using a complex potential, one can precisely specify these streamlines via the stream function. This is useful to find critical parameter values such as for the well strength parameter. As suggested, a system sensitivity analysis may quickly furnish evidence in answer to certain "what if" questions about the effectiveness or efficiency of the extraction well system after a modification of the original design.

We hope this chapter will stimulate the reader to further study hydrology in conjunction with advanced mathematical and computational analysis. The latter are important tools in more sophisticated modeling of groundwater flow, with potential implications that include efficient and effective remediation.

13.4.6 Exercises

Exercises

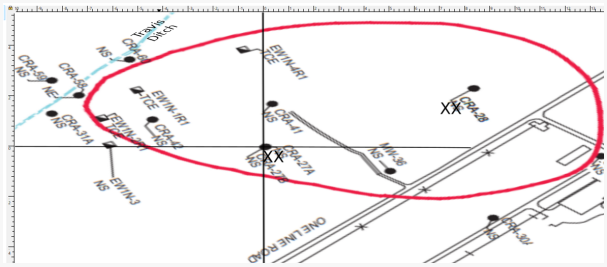
4.1 Modify $\Omega(z)$ in the JNB "Space-Leasing Plume" (available at https://github.com/wcapdat/GroundwaterFlow) so the streamline diagram for $k = .1$ tilts from northwest to southeast as shown below.



Hint: the new complex potential is $\Omega(e^{i\pi/4}z)$.

4.2 Use a stagnation point to approximate the critical value $\hat{k}$ which divides normal and inefficient systems for the basic Space Leasing plume model.

4.3 One Line North (1N) is located in a different part of the Fischer Calo superfund site. A simplified horizontal plume is shown in the Figure below. Suppose further that the two monitoring wells CRA27A located at (0,0) with potential $\Phi_1 = 708.33$ and CRA-42 located at (-4.5,1) with potential $\Phi_2 = 706.5$ show above action level contaminants. Suppose that three extraction wells EW1N-1 at (-5,2), EW1N-2 at (-6.5,1) and EW1N-4 at (-1,4) are in operation with respective strengths in a 1:5:1:1 ratio.



a) Estimate the minimum strength parameter $k = k^*$ so that the three extraction wells capture all the pollutant from the two action level monitoring wells and plot the corresponding streamlines for the flow.
b) Show that for a sufficiently small extraction strength parameter $k < k^\dagger$, upstream flow can slip in between the top and middle or middle and bottom extraction wells.

13.4.7 Supplemental JNBs

We provide several Jupyter Notebooks (JNBs) as supplementary material (downloadable from the github https://github.com/wcapdat/GroundwaterFlow) which introduce the use of Python and its libraries for the application of complex analysis to groundwater flow:

first_notebook.ipynb: Invites students to learn Python syntax by reading and correcting code, as well as generating some of their own code.

complex_numbers.ipynb: Introduces the the use of complex numbers with generic Python functionality and the functions provided by the CMath module for complex variable mathematics.

plotting_curves.ipynb: Introduces the Numpy and Matplotlib modules and demonstrates how they may be used to visualize groundwater flows using equipotentials, quiver plots, and streamlines.

Complex Potentials: Examples of sketching equipotential lines and streamlines for various complex potentials including uniform flow and flow with sinks/sources.

Rankine Oval: Used to create the figure in Section 3.6.

Space-Leasing Plume: Basic contaminant extraction model of the Fischer-Calo Space-Leasing Plume.

1NL Plume, Rotated Space-Leasing Plume: Solutions for Exercises 4.1 and 4.3.

13.4.8 References

U.S. Environmental Protection Agency. 2020. Fifth five-year review for the Fischer-Calo Superfund Site. https://semspub.epa.gov/work/05/959749.pdf

Kelly, P., Panton, R., and Martin, E.D. 1990. Three-dimensional potential flows from functions of a 3D complex variable. *Fluid Dynamics Research* 6:119–137,

Sato, K. 2015. *Complex Analysis for Practical Engineering.* Switzerland: Springer International Publishing,

13.5 Solutions to Exercises

Section 2.1

2.1.1 1) a) $1+8i$; b) $3-2i$; c) $-17+7i$; d) $\frac{1}{2}-\frac{1}{2}i$; e) $13-13i$

2.1.2 a) $e^{2x}\cos 2y + i(e^{2x}\sin 2y)$; b) $\frac{1}{2}\ln((x+1)^2+y^2) + i\arctan(\frac{y}{x+1})$; c) $(x^2-y^2-1)+i\,(2xy)$

2.1.3 If $z_1 = |z_1|e^{i\,\theta_1}$ and $z_2 = |z_2|e^{i\,\theta_2}$, then $z_1 z_2 = (|z_1|e^{i\,\theta_1})(|z_2|e^{i\,\theta_2}) = |z_1||z_2|e^{i(\theta_1+\theta_2)}$, from which we see $|z_1 z_2| = |z_1||z_2|$.

2.1.4 No, $\arg(z_1 z_2) = \arg(z_1) + \arg(z_2)$

Section 2.2

2.2.1 $\lim_{h\to 0}\frac{10(z+h)^3-10z^3}{h} = \lim_{h\to 0}\frac{10z^3+30zh^2+30z^2h+10h^3-10z^3}{h} = \lim_{h\to 0}\frac{30zh^2+30z^2h+10h^3}{h} = \lim_{h\to 0} 30zh + 30z^2 + 10h^2 = 30z^2]$

2.2.2

$\lim_{h\to 0}\frac{\overline{z+h}-\bar{z}}{h} = \lim_{h\to 0}\frac{\bar{z}+\bar{h}-\bar{z}}{h} = \lim_{h\to 0}\frac{\bar{h}}{h}$

Along the x-axis (h is real):

$\lim_{h\to 0}\frac{\bar{h}}{h} = \lim_{h\to 0}\frac{h}{h} = 1$. But, Along the y-axis ($h = ik$, k real):

$\lim_{k\to 0}\frac{\overline{ik}}{ik} = \lim_{k\to 0}\frac{-ik}{ik} = -1$.

2.2.3 Using L'Hospital's Rule, it follows that $\lim_{z\to i}\frac{z^4-1}{z^8-1} = \lim_{z\to i}\frac{4z^3}{8z^7} = \lim_{z\to i}\frac{1}{2z^4} = \frac{1}{2}$.

Section 2.3

2.3.1 $f(z) = [x+iy]^2 = x^2-y^2+i(2xy)$, so $u(x,y) = x^2-y^2$ and $v(x,y) = 2xy$. Thus $u_x = 2x = v_y$ and $u_y = -2y = -v_x$ and the CR-eqns hold with the first partials all continuous functions. Moreover, we can use CR-eqns to obtain the derivative: $f(z) = u_x + iv_x = 2x + i2y = 2z$.

2.3.2 $f(z) = x^2+y^2$, so $u(x,y) = x^2+y^2$ and $v(x,y) = 0$. Thus $u_x = 2x \neq v_y$ and the CR-eqns do not hold.

2.3.3 $f(z) = e^{x+iy} = e^x e^{iy} = e^x\cos y + ie^x\sin y$. Thus, $u = e^x\cos y$ and $v = e^x\sin y$. Moreover, $u_x = e^x\cos y = v_y$ and $u_y = -e^x\sin y = -v_x$ so the C-R eqns hold with u_x, v_x, u_y, v_y all continuous, implying that $f(z)$ is differentiable. Finally, $f'(z) = u_x + iv_x = e^x\cos y + ie^x\sin y = e^z$ as expected.

2.3.4 Let $f(z) = \log z = \ln r + i\theta$. Then $u = \ln r$ and $v = \theta$. Then $u_r = \frac{1}{r} = \frac{1}{r}v_\theta$ and $\frac{1}{r}u_\theta = 0 = -v_r$ so $f(z)$ is differentiable (except when $r = 0$). Moreover,

$$\frac{d}{dz}\log z = e^{-i\theta}[u_r + iv_r] = e^{-i\theta}\frac{1}{r} = \frac{1}{z},$$

so the same formula for real calculus holds for the complex derivative of $\log z$.

Section 2.4

2.4.1 a) Yes, $\nabla^2 u_1 = 2-2 = 0$, so u_1 is harmonic.; b) No, $\nabla^2 u_2 = 2+2 = 4 \neq 0$, so u_2 is not harmonic; c) Yes, $\nabla^2 u_3 = 0+0 = 0$, so u_3 is harmonic.; d) Yes, $\nabla^2 u_4 = e^x\cos y - e^x\cos y = 0$, so u_4 is harmonic.; e) No, $\nabla^2 u_5 = -(\cos x + \cos y) \neq 0$, so u_5 is not harmonic.

2.4.2 The harmonic conjugate of u_1 is $v_1 = 2xy + C$, the harmonic conjugate of u_3 is $v_3 = y + C$, and the harmonic conjugate of u_4 is $v_4 = e^x\sin y + C$.

Section 2.5

2.5.1 First, $\Psi = -y$. The family of lines can easily be visualized as the collection of vertical and horizontal lines in the plane, thus demonstrating that they are orthogonal.

2.5.2 a) $u(x,y) = x^2 - y^2$, $v(x,y) = 2xy$; b) Differentiating $u(x,y)$ =constant and $v(x,y)$=constant with respect to x, we obtain $\frac{du}{dx} = \frac{x}{y}$ and $\frac{dv}{dx} = -\frac{y}{x}$. Thus, the families of curves given by $u(x,y) = c_1$ and $v(x,y) = c_2$ have slopes that are opposite and reciprocal at any point of intersection, so the level curves are orthogonal.

Section 3

3.1 a) 0. There is no t dependence; b) $2x$; c) 0. Satisfies continuity equation.

Section 3.2

3.2.1 $\langle 4xe^{-y}, -2x^2e^{-y} \rangle$.

3.2.2 a) $\nabla \cdot \mathbf{V}$ =2y-2y=0. b) $\Phi = \frac{y^3}{3} - x^2y$. c) $\Phi_{xx} + \Phi_{yy} = 2y - 2y = 0$

Section 3.3

3.3.1 $\Psi_y = \Phi_x = -\frac{q_u}{A} \Rightarrow \Psi = -\frac{q_u}{A}y + t(x)$

$t'(x) = \Psi_x = -\Phi_y = 0 \Rightarrow t(x) = C$ and hence $\Psi = -\frac{q_u}{A}y + C$

$\Omega = -\frac{q_u}{A}x - i\frac{q_u}{A}y + \Phi_0 + Ci = -\frac{q_u}{A}z + C_1$. (Complex potential for uniform flow.)

The streamlines Ψ=constant are horizontal lines, and hence are perpendicular to the equipotentials Φ=constant which are vertical lines.

Section 3.4

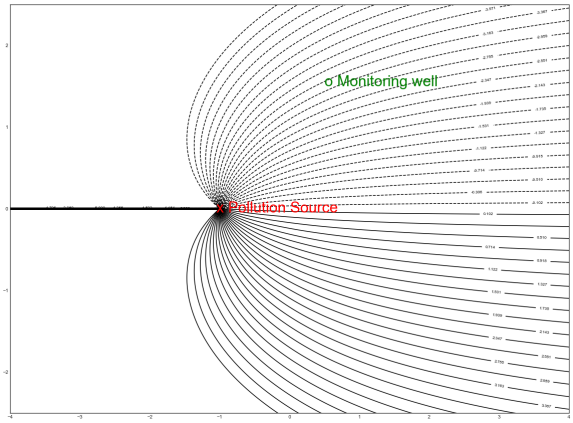
3.4.1 $\Phi = x$ and $\Psi = y$. The streamlines are horizontal (y=constant) and the flow is from right to left (higher to lower potential).

Section 3.5

3,5.1 The flow is from higher to lower velocity potential, hence from source to sink as expected.

Section 3.6

3.6.1 $\Omega(z) = -z - 2log(z+1)$ represents uniform flow with a source at $(-1, 0)$ as shown in the Figure below.



3.6.2 a) $\Psi(x,y) = -y - 4\arctan\frac{Y-1}{X+8} + 2\arctan\frac{Y-1}{X-1} - 4\arctan\frac{Y+1}{X+8} + 2\arctan\frac{Y+1}{x-1}$. b) 2 sinks and 2 sources c) The sources have twice the strength of the sinks. d) No. e) From the graph, there appears to be stagnation points at approximately (-13,0) and (3,0). f) $\Omega'(z) = -1 - \frac{4}{z-(-8+i)} + \frac{2}{z-(1+i)} - \frac{4}{z-(-8-i)} + \frac{2}{z-(1-i)} = 0$.

Section 4

4,1 See the JNB "Rotated Space-Leasing Plume" at https://github.com/wcapdat/GroundwaterFlow.

4.2 If we set $k_1 = 3k$ and turn off EWSL-4 by setting $k_2 = 0$, the complex potential becomes $\Omega(z) = k_0 z + 3k\log(z)$. We then obtain the stagnation point from

$$\frac{d\Omega}{dz} = k_0 + \frac{.3k}{z} = 0.$$

For the stagnation to be located at $z_{EWSL-4} = (10.8, 0)$, we require that $\hat{k} = \frac{-k_0(10.8)}{.3} \approx 5.22$.

4.3 See the JNB "1NL Plume" at https://github.com/wcapdat/GroundwaterFlow.

13.6 JNB LAB: Complex Potentials and Contaminant Flow

13.6.1 Complex Potentials

Introduction

In this section, we model 2-dimensional incompressible flow in various settings using the complex potential

$\Omega = \Phi(x, y) + i\Psi(x, y)$,

where $\Phi(x, y)$ is the velocity potential $\vec{V} = -\nabla\Phi(x, y)$ and $\Psi(x, y)$ is the stream function. The equipotentials are the level curves for the velocity potential: $\Phi(x, y) = constant$. The streamlines for the flow are the level curves for the stream function: $\Psi(x, y) = constant$.

We assume throughout that unless stated otherwise,

- our domain is the square with vertices (1,1),(-1,1),(-1,-1), and (1,-1).
- if there are no transformations of the plane, obstructions to the flow, or singularities such as sources or sinks, flow is in the horizontal direction.

Uniform Flow

Uniform flow means flow is in one direction (eg. in the x direction with the cross-sectional area A created by a depth h is constant). The flow rate q_u ($\frac{m^3}{s}$) and fluid velocity $V_x = q_u/A$ are constant at all points. In this case, Darcy's Law becomes

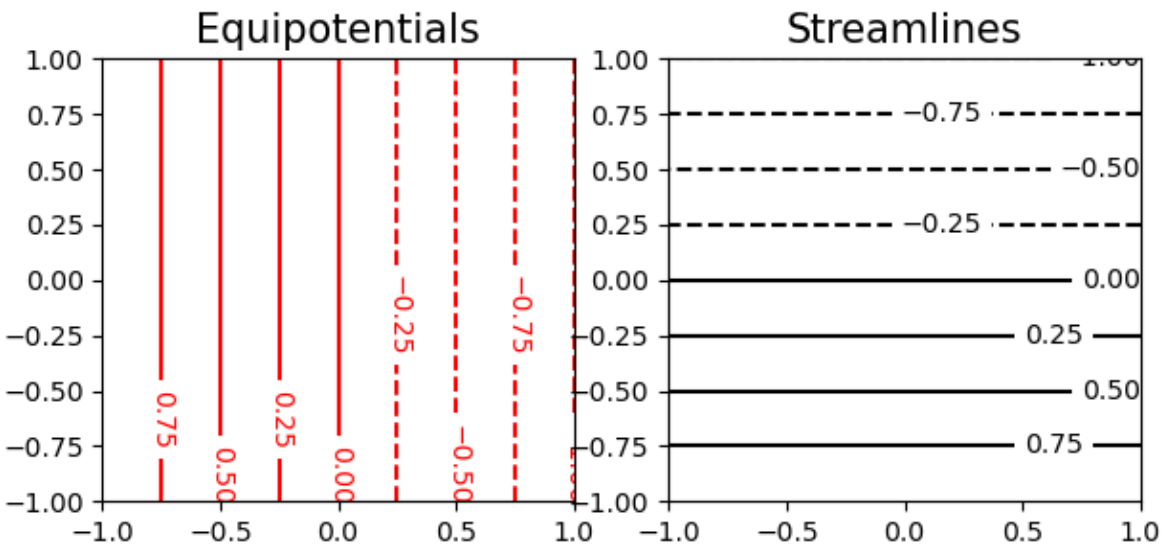
$V_x = \frac{q_u}{A} = -\frac{d\Phi}{dx}$.

Integrating, we get

$\Phi(x) = -\frac{q_u}{A}x + \Phi_0$.

For example, set $-\frac{q_u}{A} = 1$ and $\Phi_0 = 0$. Since $\psi_y(x, y) = \Phi_x = -1$, we let $\psi(x, y) = -y$ and so the complex potential is $\Omega(x, y) = -x - iy$ or $\Omega(z) = -z$.

Here is a plot of uniform flow.



Source

A source at (0,0) can be modelled by the complex potential $\Omega(z) = \log(z) = \ln(|z|) + iarg(z)$ where $-\pi < arg(z) \leq \pi$.

In this case the velocity potential is

$\Phi(z) = \ln \mid z \mid = \sqrt{x^2 + y^2}$

and the stream function is

$\Psi(z) = arg(z) = \arctan(y/x)$

Five-Spot Pattern

In the 5 point pattern we will surround the source with 4 equal strength sinks ($q_w/h = 1$) at the four corners of the domain ((1,1),(-1,1),(-1,1),(1,-1)). In this case,

- the complex potential is

$$\Omega(z) = \frac{-1}{2\pi}[\log(z) - \log(z-(1+i)) - \log(z-(1-i)) - \log(z-(-1-i)) - \log(z-(1-i))]$$

- the velocity potential is

$$\Phi(z) = \frac{-1}{2\pi}[\ln(\sqrt{x^2+y^2}) - \ln(\sqrt{(x-1)^2+(y-1)^2}) - \ln(\sqrt{(x-1)^2+(y+1)^2}) - \ln(\sqrt{(x+1)^2+(y+1)^2}) - \ln(\sqrt{(x$$

- the stream function is

$$\Psi(z) = \frac{-1}{2\pi}[\arctan\frac{y}{x} - \arctan\frac{y-1}{x-1} - \arctan\frac{y+1}{x-1} - \arctan\frac{y+1}{x+1} - \arctan\frac{y+1}{x-1}]$$

Exercises

Exercise 1

a) Make a plot of a source at (0,0) with complex potential $\Omega(z) = \log(z) = \ln(|z|) + iarg(z)$ where $-\pi < arg(z) \leq \pi$.

b) Make a plot for one quarter of a 5-spot pattern which consider the terms in the potential for just the source at (0,0) and a sink at (1,1):

$\Omega(z) = \frac{-1}{2\pi}[\log(z) - \log(z - (1 + i))]$

Hint: The stream function, which accounts for the choice of branch cuts (negative x-axis at the branch point (0,0) and in the direction of the positive x-axis at the branch point (1,1), can be evaluated as

$\Psi(x, y) = \frac{-1}{2\pi}[\pi + \arctan(\frac{y}{x})] + \frac{1}{2\pi}[2\pi - \arccos(\frac{X-1}{\sqrt{(X-1)^2+(Y-1)^2}})]$

13.6.2 Fischer-Calo Contaminant Model

Setting the Scene

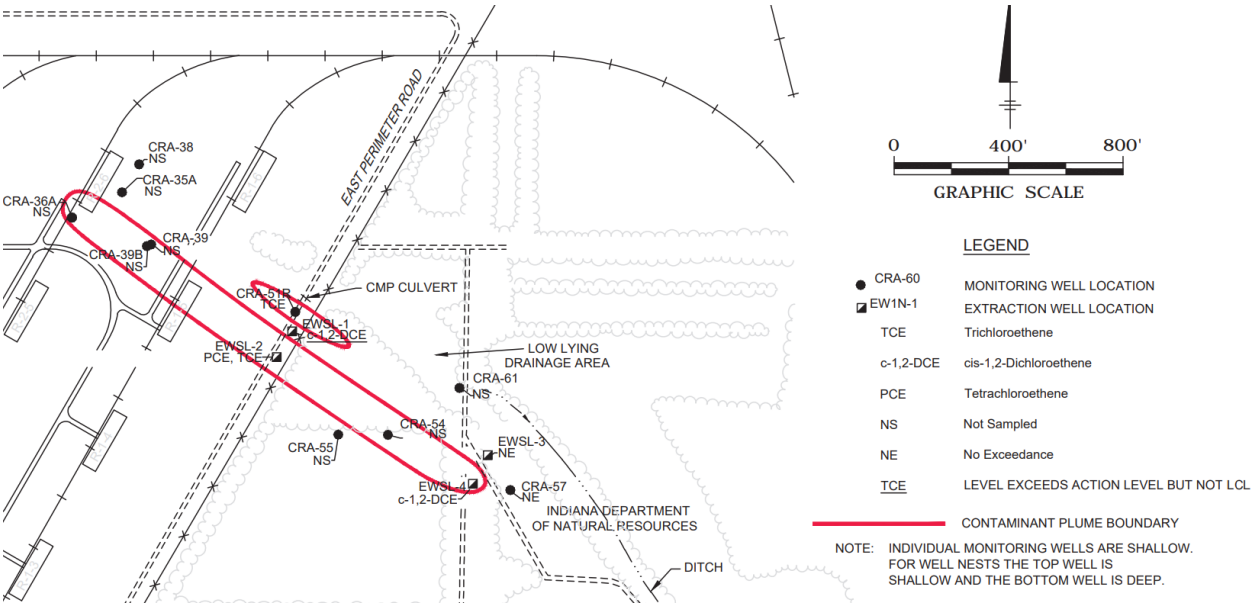
The Fisher-Calo Superfund Site is located in the multiple square-mile Kingsbury Industrial Development Park (KIDP) in La Porte County, Indiana. Two communities, Kingsbury and Kingsford Heights are located within 2 miles of KIDP. An industrial chemical processing and distribution facility operated on that site. Fisher-Calo Chemicals and Solvent Company and various subsidiaries began operations at KIDP in the early 1970s, with wastes buried on-site in thousands of drums. Spills, leakage, and a large fire in March 1978 resulted in the contamination of soil and groundwater with polychlorinated biphenyls (PCBs) and volatile organic compounds (VOCs). The contamination has spread from the source due to groundwater flow, and thus the contaminant source is not a mathematical source per se. The site's long-term remedies include:

```
- excavation, consolidation, and offsite disposal of 3,600 drums (1994-1996);
- soil source area remediation via excavation and off site disposal of PCB  impacted␣
 ↪soils, soil vapor extraction (SVE), and bio-venting of semi-volatile organic␣
 ↪compound (SVOC) impacted soils, and SVE of volatile organic compound (VOC) impacted␣
 ↪soils (1996-2001); and
- groundwater extraction and treatment (1998-present) with partial Pilot Study␣
 ↪evaluation of enhanced monitored natural attenuation (2020-present).
```

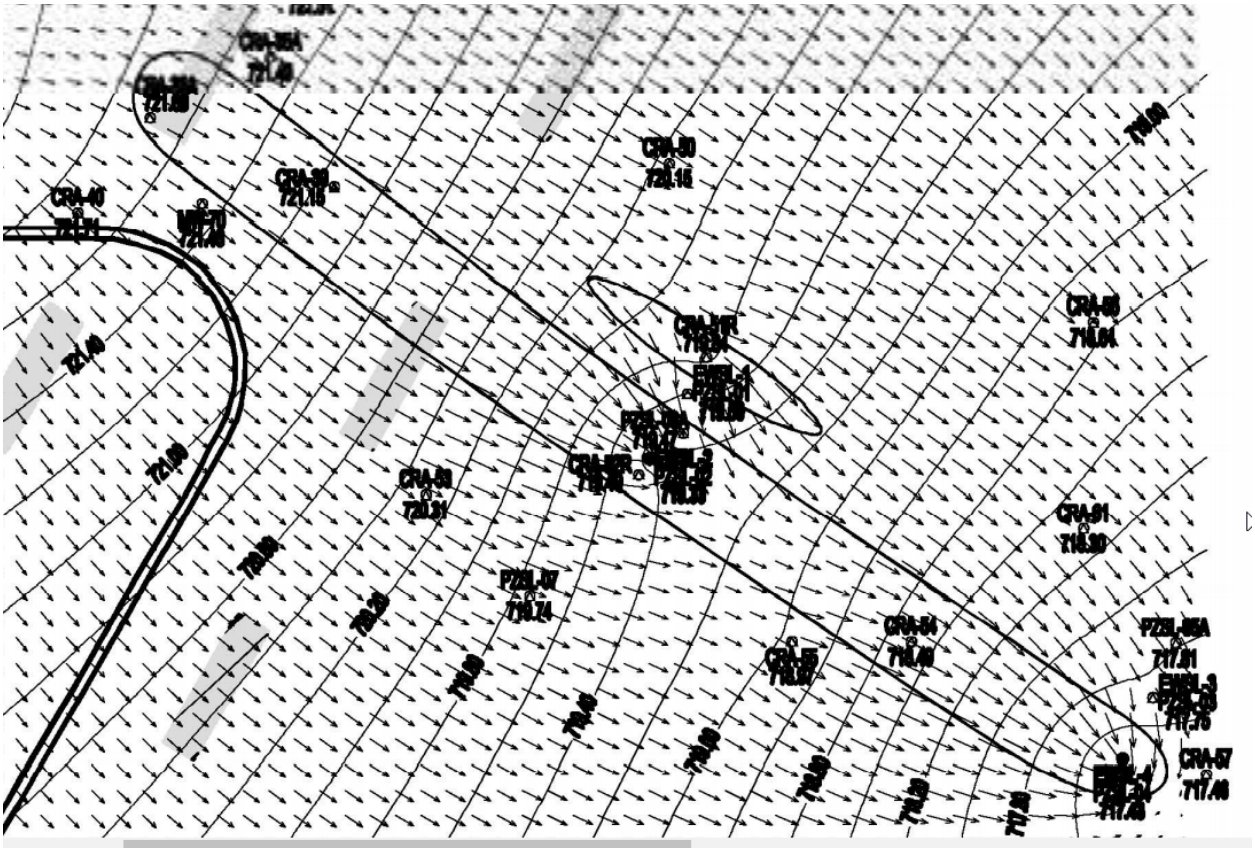
The EPA conducts 5 year reviews to monitor and evaluate adherence to planned operations.

Math modeling was used to support the goal to design an effective contaminant extraction well remediation system. Such a system includes both pollutant extracting wells (EW) and plume monitoring wells (MW), and is the traditional remediation approach. The Fischer-Calo Groundwater Extraction and Treatment System consisting of 16 extraction wells (EWs) began operating in Feb 1998. More than four different contaminated groundwater plumes were identified. We focus on one of these plumes, called the Space Leasing plume.

A rough approximation of the main Fisher-Calo Space Leasing plume boundary is shown encircled in the long bold oval. For this plume, one extraction well (EWSL-2) is located mid-plume, and a second extraction well (EWSL-4) downstream at the plume's bottom right.



Groundwater direction arrows are orthogonal to the lines of equipotential. Figure Credit: [U.S. Environmental Protection Agency 2020]



Groundwater at Space Leasing is present in the shallow, unconfined sand aquifer with a saturated thickness of approximately 48 to 58 feet. The direction of groundwater flow is to the southeast at an approximate horizontal hydraulic gradient of 0.002 ft/ft under 2019 pumping conditions. Groundwater in the Space-Leasing plume is impacted with chlorinated volatile organic compounds (CVOC), including trichloroethene (TCE), tetrachloroethene (PCE), and cis-1,2-Dichloroethene (cis-1,2-DCE) at concentrations greater than action levels in the upper half of the unconfined aquifer.

The extent of the CVOC plume at Space-Leasing is approximately 150 feet wide by approximately 1,750 feet long and is oriented in the northwest to southeast direction.

Our model is based on 2019 data for the Space Leasing Plume remediation [U.S. Environmental Protection Agency 2020] which involved a pump and treat system with two extraction wells, one mid-plume (EWSL-2), and the other at the down-stream end (EWSL-4). We considered the following modeling problem:

```
Assess the containment of above action level contaminant at monitoring wells by the␣
↪two extraction wells (EWSL-2, EWSL-4).
```

Our model utilizes the beauty and ingenuity of complex variables to analyze 2-dimensional groundwater flow. Sato's wonderful book Complex Analysis for Practical Engineering [Sato 2015] provides complete background for our approach.

Model

We model the Space-Leasing Plume's pollution extraction system using a complex potential of the form

$$\Omega(z) = k_0 z + k_1 \log(z - z_{EWSL-2}) + k_2 \log(z - z_{EWSL-4}) + C.$$

where $x_{EWSL-2} = 0$ and $x_{EWSL-4} = 10.8$ are the positions of the two extraction wells located on a horizontal axis. $C = \Phi_0$ is a real constant. We determine the coefficients as follows:

Based on the equipotential map for the SL plume, we set the baseline potential at $\Phi_0 = 717$, so $C = 717$.

Since the strengths of the mid-plume and downstream extraction wells are in a 3:2 ratio, we let $k_1 = 3k$ and $k_2 = 2k$. We call k the extraction well strength parameter. It follows that the velocity potential has the form

$$\Phi(z) = \Phi(x + iy) = k_0 x + +3k \ln(\mid z - z_{EWSL-2} \mid) + 2k \ln(\mid z - z_{EWSL-4} \mid) + 717.$$

Under the assumption of constant uniform flow, if we estimate k_0 using two vertical equipotentials, one at $x = x_1$ with value Φ_1 and one at $x_1 + d$ with value Φ_2, then $k_0 x_1 = \Phi_1$and $k_0(x_1 + d) = \Phi_2 \Rightarrow k_0 \approx \frac{\Phi_2 - \Phi_1}{d}$. Using process on three equipotentials and averaging, we obtained the estimate $k_0 \approx -.145$.

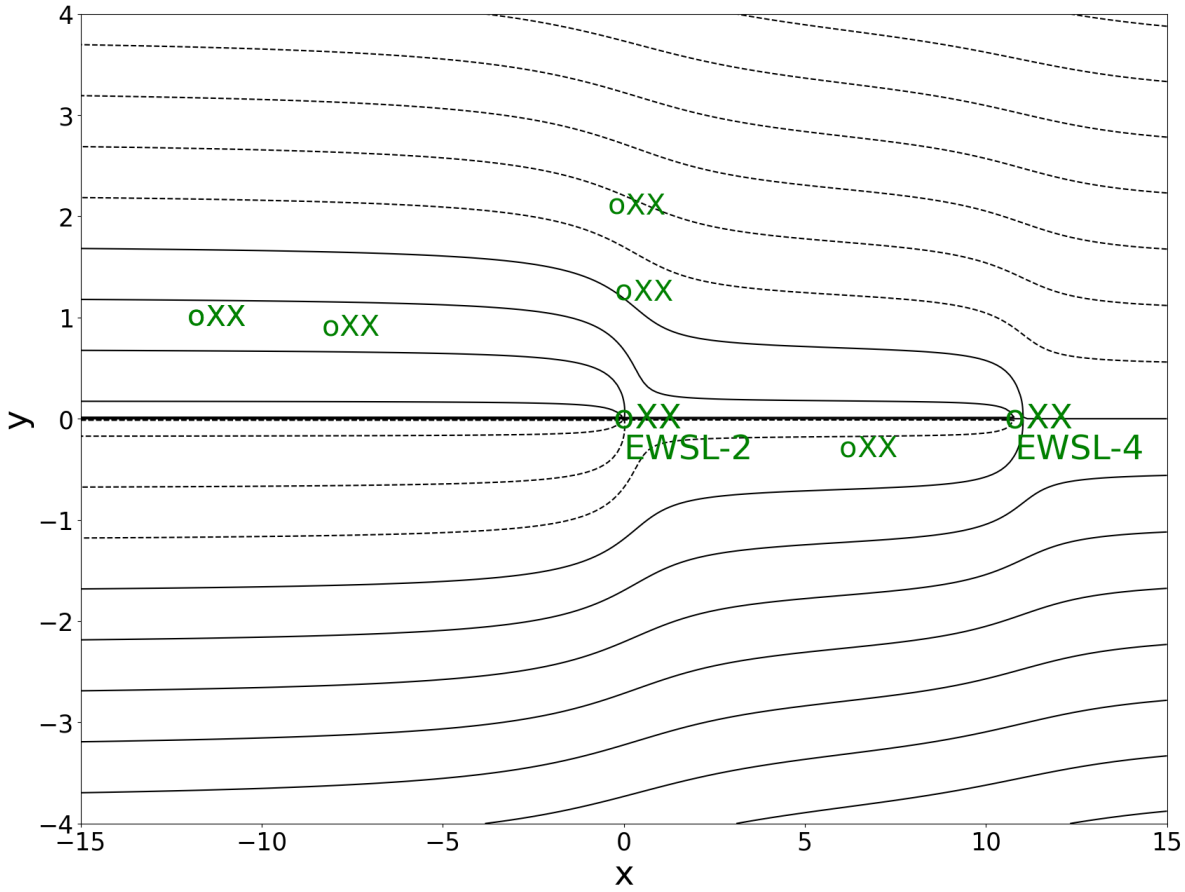
We thus assume that $\Phi(z) = \Phi(x + iy) = -.145x + 3k \ln(\mid z \mid) + 2k \ln(\mid z - 10.8 \mid) + 717$,

Due to the nature of approximating the values of k_0 and k, we perform a sensitive analysis that reveals the different types of behavior that can be observed for various choices of k.

Case 1: Ineffective System

$(k = k_{ave} \approx .16)$

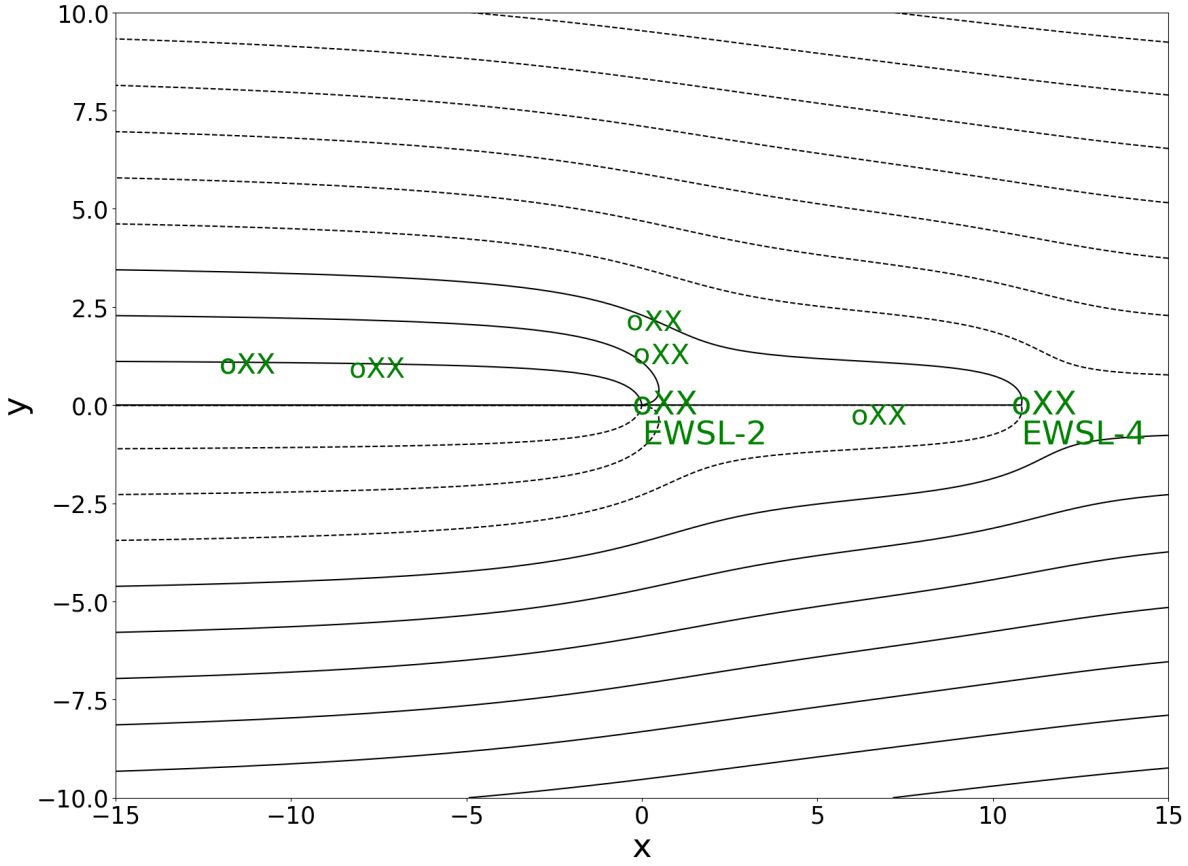
In this case, the well-extraction system does not capture all the streamlines emanating from above action level positions.



Case 2: Regular System

$(k \approx .4)$

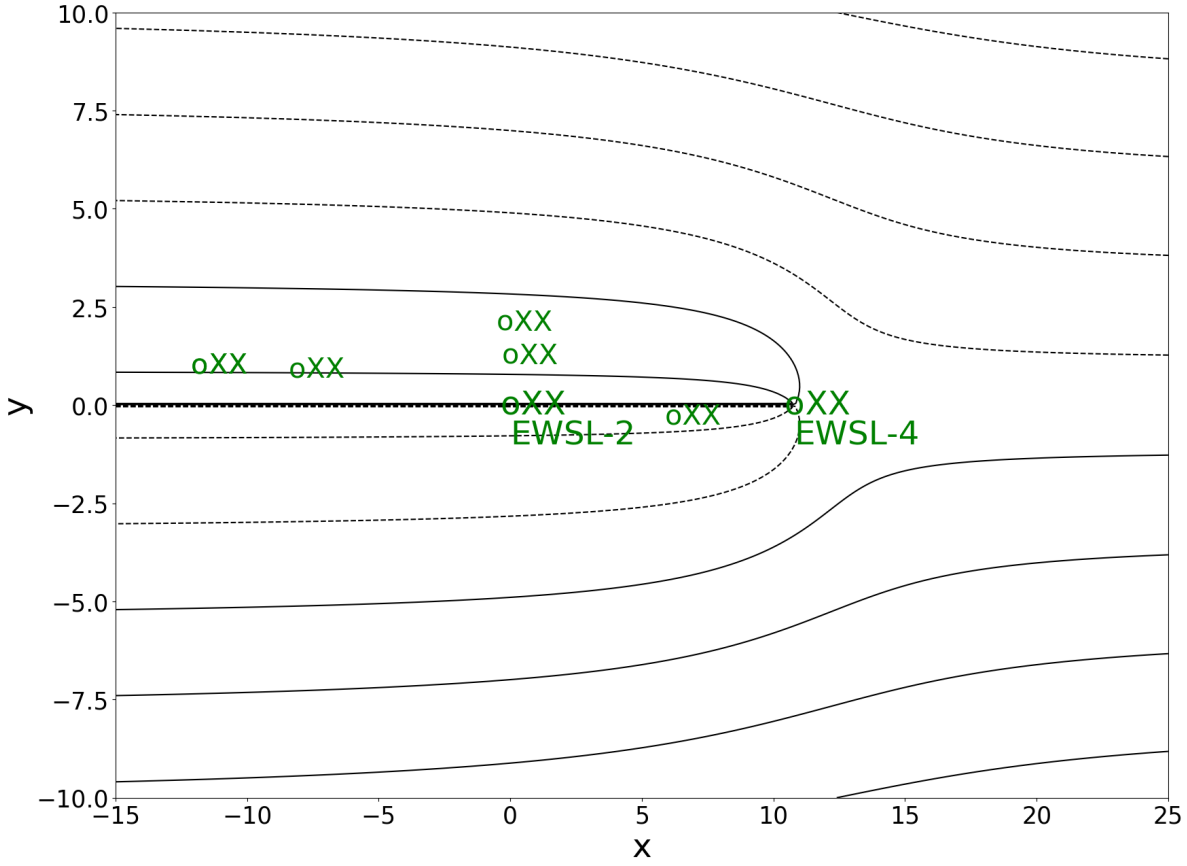
In this case, both the mid-plume well (EWSL-2) and the downstream extraction well EWSL-4 capture streamlines emanating from above action level positions. All targeted pollutant is captured by the 2 wells.



Case 3 Inefficient System

$(k \approx 1)$

In this case, the down-stream well (EWSL-4) can capture all the streamlines emanating from above action level positions rendering the mid-plume well (EWSL-2) unnecessary (the strength of the downstream well EWSL-2 is reduced to 0 in order to show the streamlines that can be captured by EWSL-4).



Exercise

Exercise 2 (Sensitivity Analysis)

a) At what point does the 3k:2k system become ineffective?

b) At what point does the 3k:2k system become inefficient?

c) How does reversing the strength ratio to 2k:3k affect the critical value $k = k^*$ at which the system becomes ineffective?

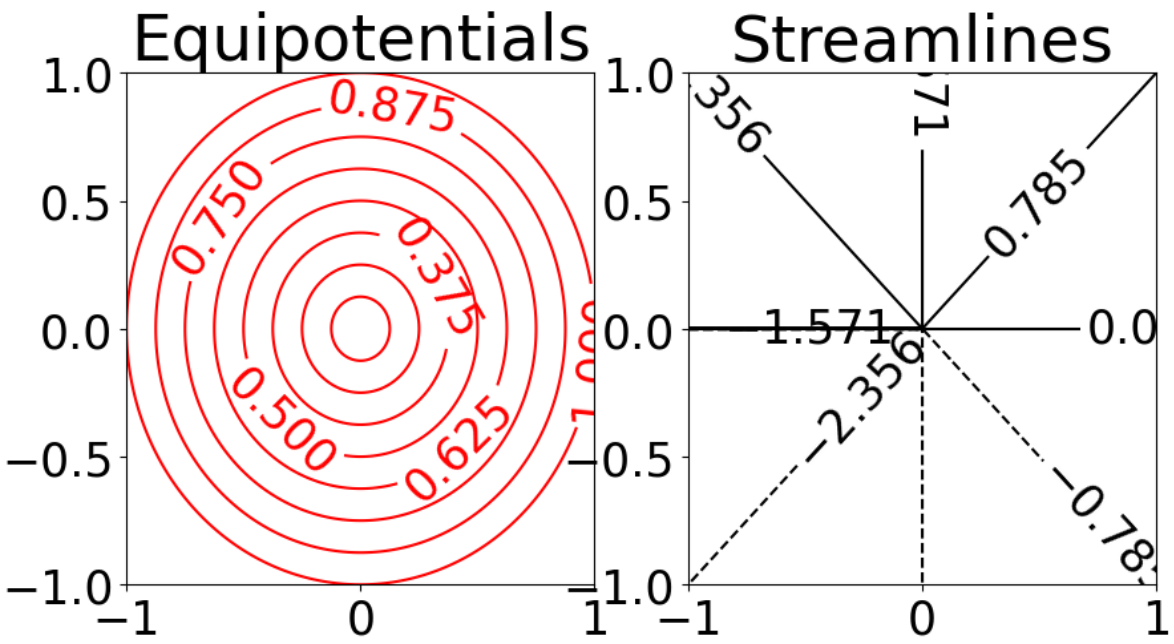
d) How does reversing the strength ratio to 2k:3k affect the critical value $k = \hat{k}$ at which the system becomes inefficient?

13.7 JNB Lab Solutions

13.7.1 Complex Potentials

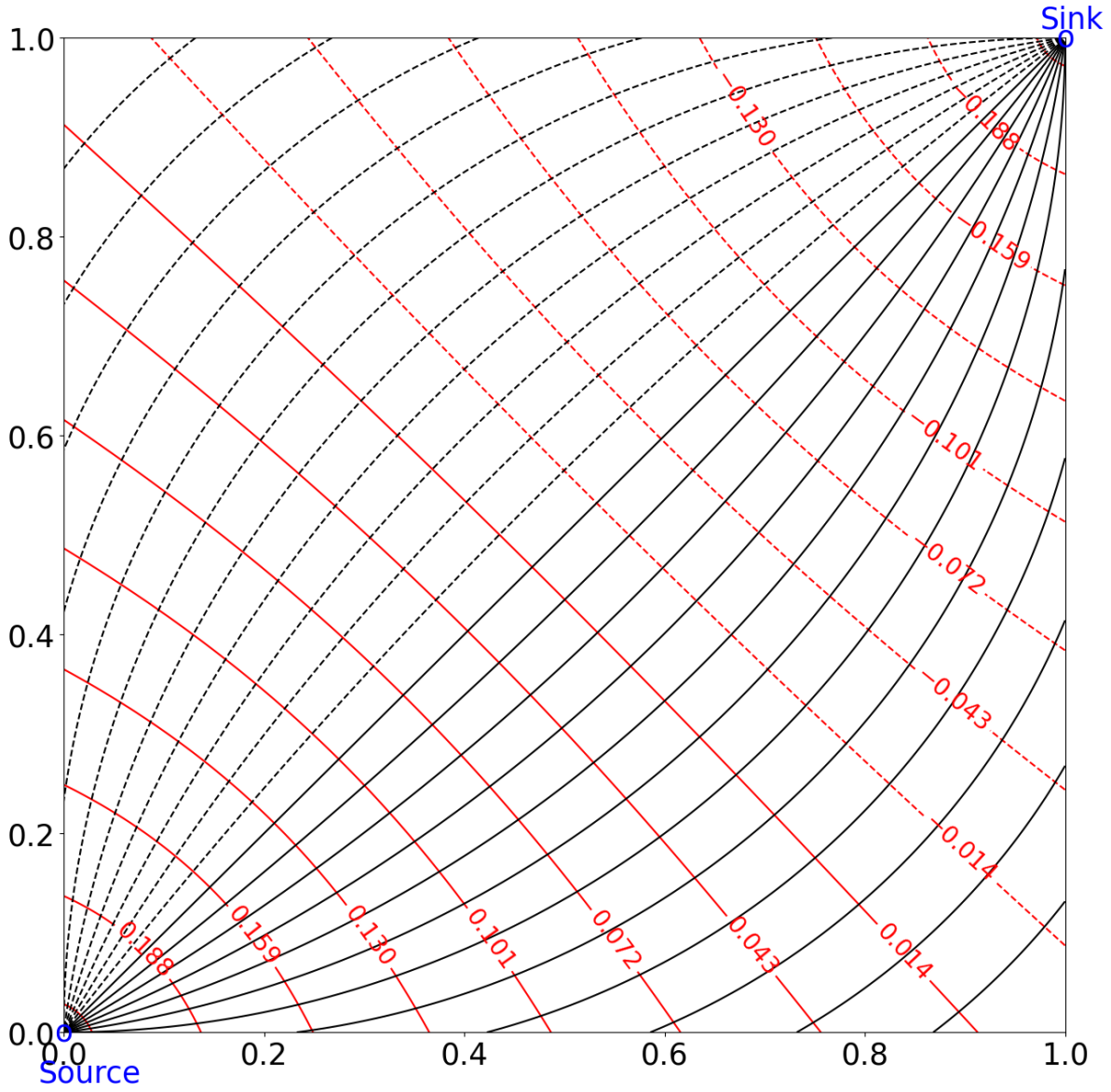
Solution to Exercise 1

a)



b)

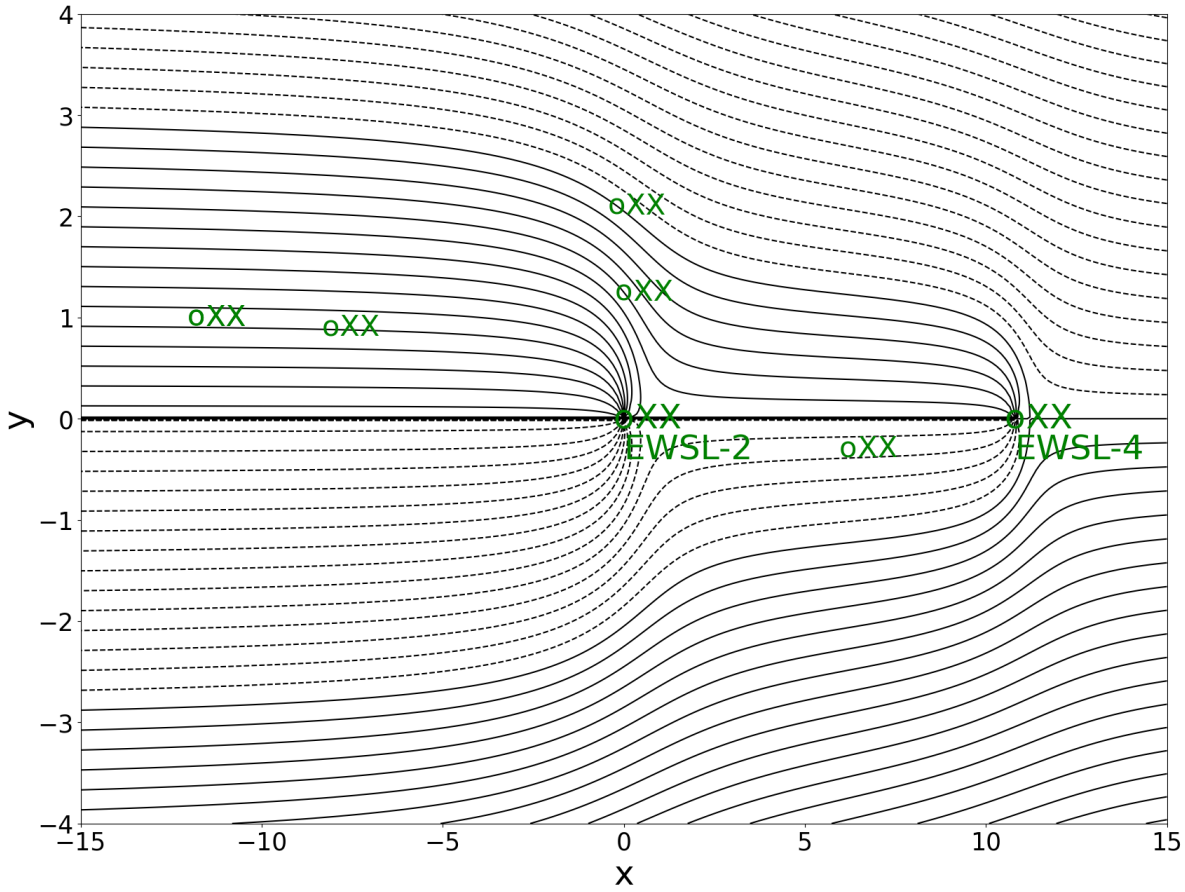
```
C:\Users\pisihara\AppData\Local\Temp\ipykernel_11044\3572428666.py:14:␣
↪RuntimeWarning: invalid value encountered in divide
  Z1 =  (-1/(2*np.pi))*(np.pi+np.arctan2(Y,X))+(1/(2*np.pi))*(2*np.pi-np.arccos((X-
↪1)/np.sqrt((X-1)**2+(Y-1)**2)))
```



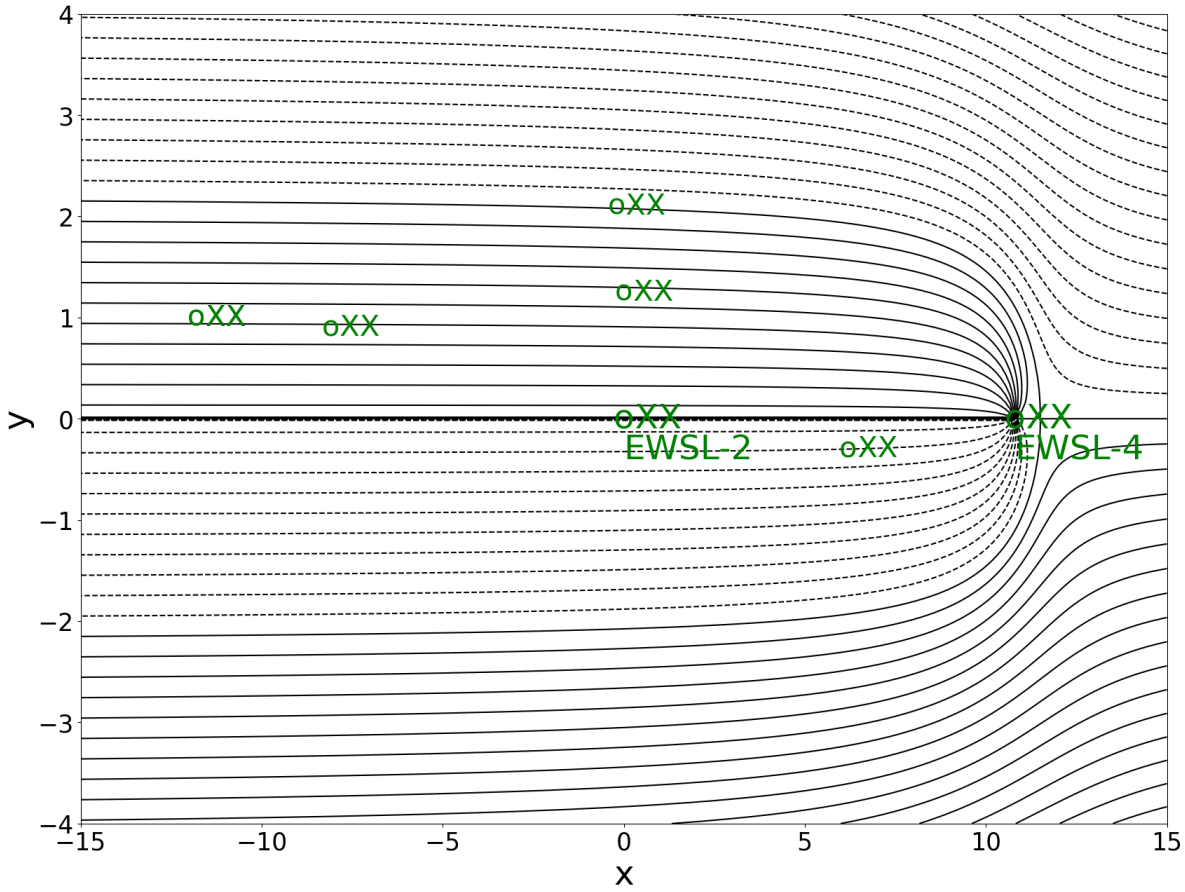
13.7.2 Fischer-Calo Contaminant Model

Solution to Exercise 2

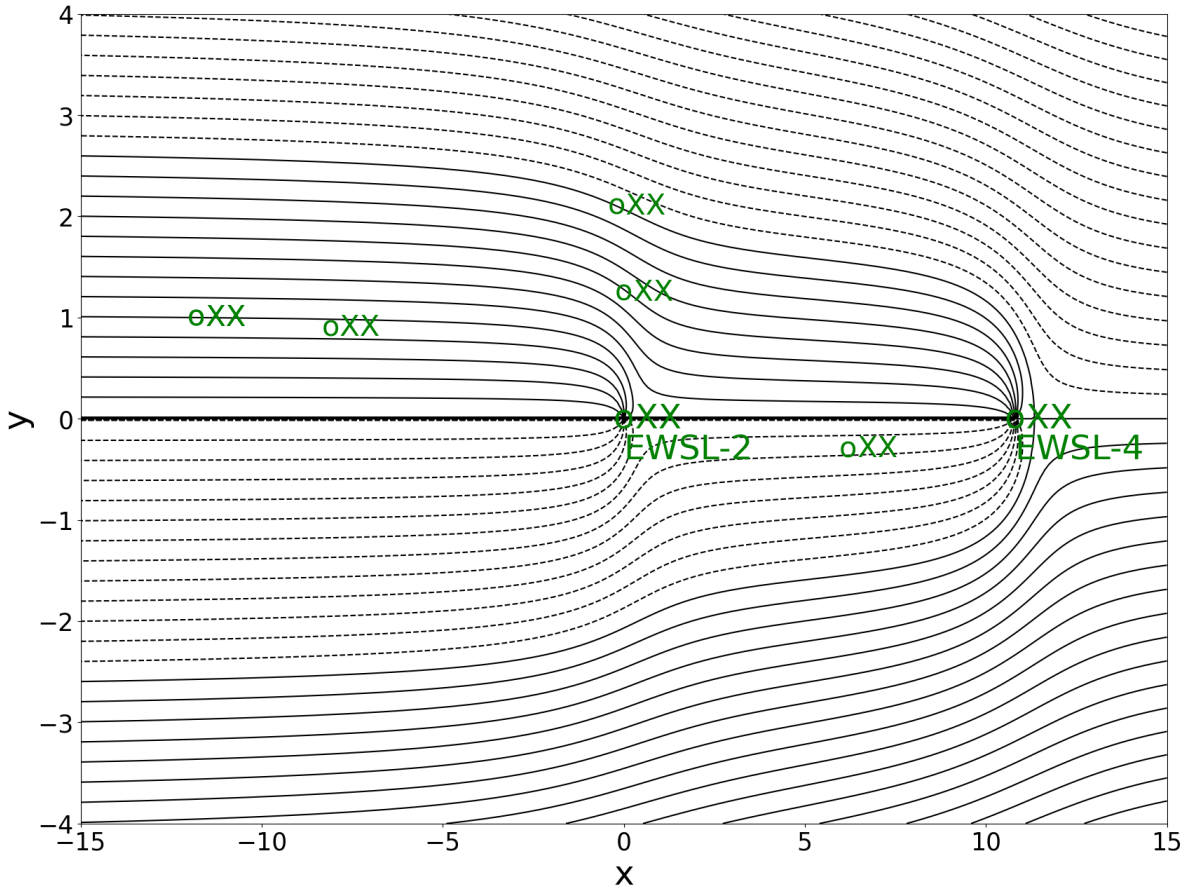
a) $k = k^* \approx .28$



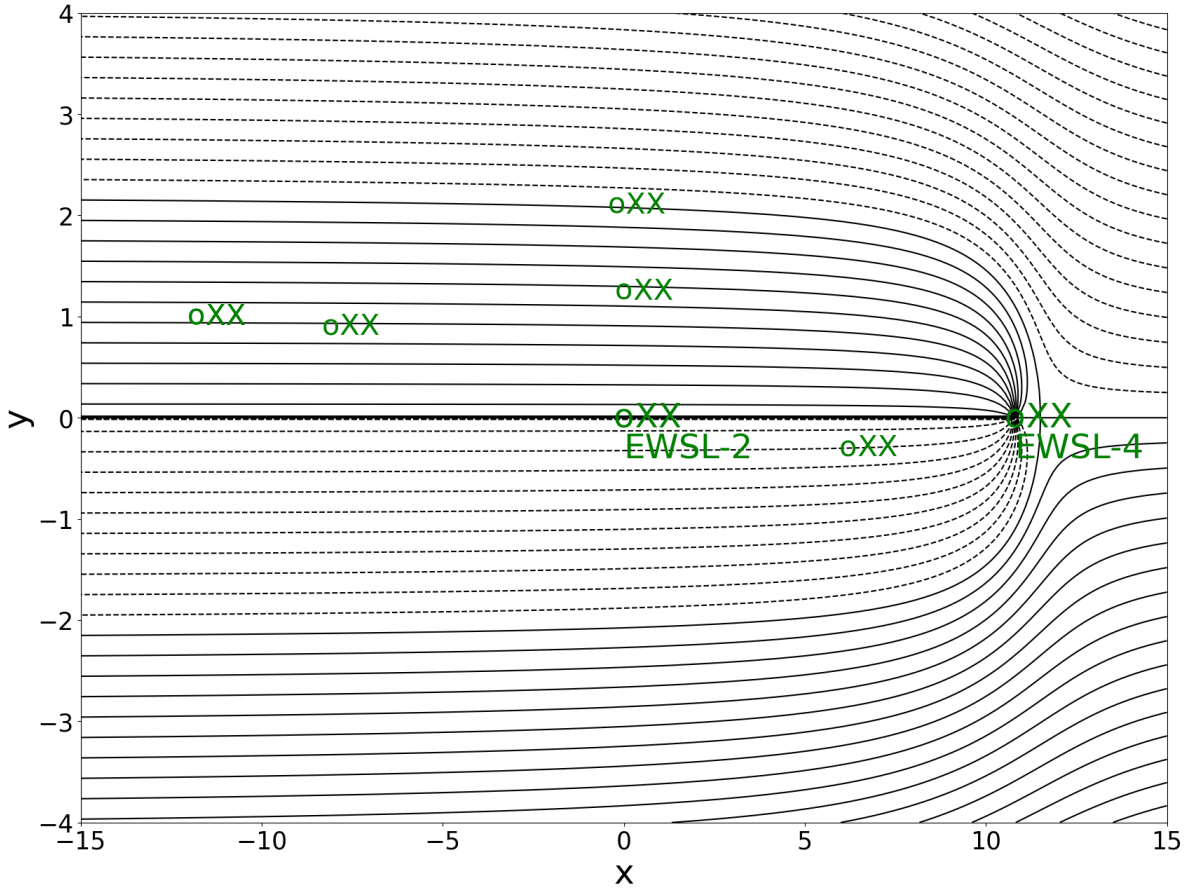
b) $k = \hat{k} \approx .51$



c) It reduces it to $k^* \approx .25$



d) It reduces it to $\hat{k} \approx .34$ (Note: This can be obtained from $(.51)(2/3)$)


oXX
oXX
oXX
oXX
oXX
oXX
oXX
EWSL-2
EWSL-4
y
x
4
3
2
1
0
−1
−2
−3
−4
−15
−10
−5
0
5
10
15

CHAPTER 14

ADVANCED DATA ANALYSIS - Jonathan Zhu, Claire Wagner, Ying Li

Note

Jonathan Zhu, Claire Wagner, and Dr. Li will discuss advanced methods in analyzing U.N. Sustainable Development Goal data.

14.1 Introduction

The dataset used in the exercises for this section is from a citizen science project. It consists of 40067 text excerpts labeled with 16 United Nations Sustainable Development Goals (UN SDG) categories and can be found at https://zenodo.org/record/7540165#.ZAF10uzMKfU. As of now, all the exercises use the 2023.01 version of the dataset, which is the version used when designing the exercises and the content for this section. Please be aware that the data file format has gone through significant changes since the first version, 2021.09.

The UN SDG framework is a UN 2030 thematic framework for world development. Adopted in 2015, it has 17 comprehensive goals, with 169 targets and 232 indicators. The UN SDGs are universal, being accepted and adopted by countries all over the globe, and represents a proxy of an ideal future.

SUSTAINABLE DEVELOPMENT GOALS
17 GOALS TO TRANSFORM OUR WORLD
1 NO POVERTY
2 ZERO HUNGER
3 GOOD HEALTH AND WELL-BEING
4 QUALITY EDUCATION
5 GENDER EQUALITY
6 CLEAN WATER AND SANITATION
7 AFFORDABLE AND CLEAN ENERGY
8 DECENT WORK AND ECONOMIC GROWTH
9 INDUSTRY, INNOVATION AND INFRASTRUCTURE
10 REDUCED INEQUALITIES
11 SUSTAINABLE CITIES AND COMMUNITIES
12 RESPONSIBLE CONSUMPTION AND PRODUCTION
13 CLIMATE ACTION
14 LIFE BELOW WATER
15 LIFE ON LAND
16 PEACE, JUSTICE AND STRONG INSTITUTIONS
17 PARTNERSHIPS FOR THE GOALS
SUSTAINABLE DEVELOPMENT GOALS

Each text in the dataset has been labeled with a corresponding SDG and then shown to volunteers, who were asked to evaluate whether or not they thought the SDG was a good match for the text.

We recommend creating a new Python environment for this section; you can do this in the terminal by running the following code chunk:

```
conda deactivate
conda create -n nlp
conda activate nlp
conda install -c conda-forge spacy
conda install -c conda-forge scikit-learn
conda install ipykernel
conda install seaborn
conda install nltk
```

The libraries for this section include `pandas`, `scikit-learn`, `nltk`, `spaCy`, and `tensorflow`, as well as some sub-packages in these libraries. Code to import the necessary libraries will be provided in each section. For example, the following code imports the `pandas` library.

```
import pandas as pd
```

Most importantly, it is necessary to download the full dataset and put it in a directory you can access. At the beginning of each section, you should modify the first code cell to use this directory:

```
# change this to your own data directory
data_dir = "data/"
```

We can then load the dataset and examine the first few rows as follows:

```
text_file_name = "osdg-community-data-v2023-01-01.csv"
text_df = pd.read_csv(data_dir + text_file_name,sep = "\t",  quotechar='"')
col_names = text_df.columns.values[0].split('\t')
```

(continues on next page)

(continued from previous page)

```
text_df[col_names] = text_df[text_df.columns.values[0]].apply(lambda x: pd.
↪Series(str(x).split("\t")))
text_df = text_df.astype({'sdg':int, 'labels_negative': int, 'labels_positive':int,
↪'agreement': float}, copy=True)
text_df.drop(text_df.columns.values[0], axis=1, inplace=True)
text_df.head()
```

```
                          doi                              text_id  \
0  10.6027/9789289342698-7-en  00021941702cd84171ff33962197ca1f
1        10.18356/eca72908-en  00028349a7f9b2485ff344ae44ccfd6b
2  10.1787/9789264289062-4-en  0004eb64f96e1620cd852603d9cbe4d4
3     10.1787/5k9b7bn5qzvd-en  0006a887475ccfa5a7f5f51d4ac83d02
4  10.1787/9789264258211-6-en  0006d6e7593776abbdf4a6f985ea6d95

                                                text  sdg  labels_negative  \
0  "From a gender perspective, Paulgaard points o...    5                1
1  Labour legislation regulates maximum working h...   11                2
2  The average figure also masks large difference...    3                1
3  The extent to which they are akin to corruptio...    3                1
4  A region reporting a higher rate will not earn...    3                2

   labels_positive  agreement
0                8   0.777778
1                1   0.333333
2                8   0.777778
3                2   0.333333
4                2   0.000000
```

The features of the dataset that are relevant for our purposes are as follows:

- `text` - an excerpt from the document
- `sdg` - the SDG assigned to the text
- `labels_negative` - the number of volunteers who thought that the assigned SDG did not match the text
- `labels_positive` - the number of volunteers who thought that the assigned SDG did match the text
- `agreement` - agreement score, calculated as $agreement = \frac{|labels_{positive} - labels_{negative}|}{labels_{positive} + labels_{negative}}$

As the sample entries above show, the SDGs in the dataset are denoted by numbers. For more human-readable names and definitions of the SDGs, you can download the file `sdg_names_definitions.csv` from the repository. Put this file in the same data directory as the dataset.

```
file_name = "sdg_names_definitions.csv"
sdg_names = pd.read_csv(data_dir + file_name)
sdg_names.head()
```

```
   sdg                    sdg_name  \
0    1                  No Poverty
1    2                Zero Hunger
2    3  Good Health and Well Being
3    4           Quality Education
4    5             Gender Equality

                                 sdg_definition
0           End poverty in all its forms everywhere
```

(continues on next page)

(continued from previous page)

```
1  End hunger, achieve food security and improved...
2  Ensure healthy lives and promote well-being fo...
3  Ensure inclusive and equitable quality educati...
4  Achieve gender equality and empower all women ...
```

This provides the framework for the rest of the examples and exercises we will use throughout this section.

```
# import libraries
import pandas as pd
import spacy
import nltk
nltk.download('punkt', quiet=True) # download punkt (if not already downloaded)
nltk.download('stopwords', quiet=True) # download stopwords (if not already␣
↪downloaded)

# change this to your own data directory
data_dir = "data/"

# read and preprocess data
text_file_name = "osdg-community-data-v2023-01-01.csv"
text_df = pd.read_csv(data_dir + text_file_name,sep = "\t",  quotechar='"')
col_names = text_df.columns.values[0].split('\t')
text_df[col_names] = text_df[text_df.columns.values[0]].apply(lambda x: pd.
↪Series(str(x).split("\t")))
text_df = text_df.astype({'sdg':int, 'labels_negative': int, 'labels_positive':int,
↪'agreement': float}, copy=True)
text_df.drop(text_df.columns.values[0], axis=1, inplace=True)
```

14.2 Preprocessing

Natural language processing (NLP) refers to the use of computers to process, analyze, and interpret human languages. NLP often involves the use of machine learning (ML), which is explained in more detail in *section 5*. Usually, natural language processors operate on both speech data and textual data. There are several components necessary to understand the structure and meaning of human language.

From a linguistics perspective, it is important to look at the following:

- **syntax**, the actual rules that define how words are combined to form understandable sentences;
- **semantics**, referring to the meaning behind the phrases and sentences formed by syntax;
- **morphology**, referring to the ways in which words can take different forms.

Then, from a computational perspective, we can take the rules from these linguistics aspects and transform linguistic knowledge into rule-based and/or ML-based algorithms to solve problems related to natural language processing.

Exercise 1

Explore the Lexalytics NLP demo found at https://www.lexalytics.com/nlp-demo/. Choose any of the provided packs and look at the provided sample texts.

- Look under Document and Themes. What words and phrases does the model highlight for each sample?
- Look under Topics. What topics does the model detect in each sample? How accurate and comprehensive is its list?

14.2.1 Tasks

Within the realm of preprocessing textual data, we first must perform certain detections and tasks; this ensures that we have textual data that make sense and are meaningful. The following all help at ensuring data cleanliness:

- **Syntactic analysis:** Does the grammar make sense? Is the text grammatically correct?
- **Semantic analysis:** Are we aware of the meaning of the words in context? Do we understand the structure, word interaction, and related concepts of the text?
- **Keyword extraction:** What are the most important words in the text?
- **Named entity resolution:** How do we identify and extract entities (names, organizations, addresses and places) and relationships?
- **Text classification:** Can we organize our text into predefined categories? Ways of doing this include:
  - **Sentiment analysis:** Can we classify text (such as customer feedback) into positive and negative feedback?
  - **Email filtering:** If we are taking our texts from emails, can we classify email text as spam mail and remove spam?
  - **Intent detection:** What is the text generator trying to achieve? For example, searching "apple" could indicate an intent of buying, eating, or researching apples.
  - **Language detection:** Can we classify the body of text into languages, with an associated probability?

Exercise 2

Take another look at the Lexalytics NLP demo found at https://www.lexalytics.com/nlp-demo/. Choose any of the provided packs and look at the provided sample texts.

- Look under Document and Themes, which present the model's sentiment analysis. What sentiment scores does the model give to the various words and phrases it has extracted? Do you agree or disagree with its verdicts?

Looking at the UN SDG dataset, we find that it includes positive or negative labels. Since each text is classified with a single goal, positive labels indicate the agreement of that text with the goal, while negative labels indicate disagreement. We use `sum()` to sum the negative labels and the positive labels:

```
text_df.labels_negative.sum() + text_df.labels_positive.sum()
```

```
277524
```

Compare this to the number of text samples in the dataset:

```
len(text_df)
```

```
40062
```

You may notice that the sum of the positive and negative labels does not equal the number of texts in the dataset. This is because the texts have been labeled by more than one volunteer. As the table below shows, between 3 and 990 volunteers labeled each text, with an average (and median) of about 7 volunteers per text. The columns `labels_positive` and `labels_negative` give the total number of positive and negative labels assigned to each text by these volunteers.

```
(text_df['labels_positive'] + text_df['labels_negative']).describe(percentiles=[0.5])
```

```
count    40062.000000
mean         6.927363
std         15.578931
min          3.000000
50%          7.000000
max        990.000000
dtype: float64
```

With this in mind, determining a text's agreement with the assigned SDG goal will be more complicated than simply reading the positive and negative labels. These observations thus provide some beginning rationale for the necessity of preprocessing.

The data preprocessing we will need to perform includes some tasks specific to textual data, but it also includes some work that can be done on datasets more generally. For our UN SDG dataset, it is useful to remove the columns that we don't need, which for this dataset are any columns with N/A values in all entries. We can use the following code to do so:

```
# save the original number of columns in a variable
ncols_old = text_df.shape[1]
# remove columns comprised solely of N/A values
text_df.dropna(axis=1, how='all', inplace=True)
print("Number of columns removed:", ncols_old - text_df.shape[1])
```

```
Number of columns removed: 0
```

In this case, there were no columns comprised solely of N/A values.

It is also a good idea to check our data for any other discrepancies, such as missing or incorrect entries.

Exercise 3

- Modify the code above to remove any rows that contain only N/A values. (Hint: change the value of the `axis` parameter.) How many rows, if any, were removed?
- Using the `isna()` and `any()` functions, check the data for any remaining N/A values. Is it a good idea to remove entries with N/A values?

For our UN SDG dataset, we want texts that are clearly classified into a single goal. Positive labels indicate agreement with the labeled goal, while negative labels indicate disagreement. With this in mind, our filtering will remove the rows with labeling agreement less than or equal to 0.5, as well as the rows where the number of positive labels minus the negative labels is less than or equal to 2.

```
text_df = text_df.query("agreement > 0.5 and (labels_positive - labels_negative) > 2")
text_df.reset_index(inplace=True, drop=True)
```

Exercise 4

Use `text_df.info()` to examine information about the dataset after running the preprocessing in this section. What columns are included? What data types are there, and how many entries? Does this seem like a reasonable size of data to work with?

14.2.2 Levels of Processing

In (almost) any language, there are distinct levels of a text we can focus on: **characters** are strung together to form **words**, which are put together into **sentences**, then **paragraphs**, **documents**, and a whole set of documents, or a **corpus**. In all these cases, we put multiples together to make the next level, and decisions in preprocessing at one level will inherently impact the next level.

We will not be focusing on using algorithms to process whole documents or anything above that level, so we can look at preprocessing at the remaining levels: characters, words, sentences, and paragraphs.

For characters, we typically process textual data by removing special characters, punctuations, and normalizations. However, this is not necessarily the case for every NLP approach. In some cases, taking away these special characters, punctuations, and normalizations will completely alter the meaning of higher levels of text.

For words, we first have to define words within our text snippets. This is known as **segmenting**. In the English language, we can use white space to segment text into words. However, this is not always the case with other languages, so it is important to know the linguistic rules of the primary language you are working with. Additionally, we perform some removals and/or replacements for odd words such as abbreviations, acronyms, numbers, and misspelled words. Finally, we normalize our text data; this is typically done by **stemming** (covered later in this section) or **lemmatization**.

For sentences, we first have to define sentence boundaries. In most languages, including English, sentences are started with a capital letter and ended with a period. However, periods and capital letters are also used within sentences, and capital letters or periods might not even exist in other languages, which again highlights the importance of knowing the language's rules. Within sentences, we can also mark phrases. In English sentences, we can typically label phrases in sentences with a subject, predicate, and object; the subject is the one performing the action, or predicate, on or directed towards the object. Finally, we parse the sentence by tagging the words with their respective part of speech. Note that this can only be done on the sentence level, not the word level, as the same word can have a different part of speech depending on how it is used in a sentence.

Finally, **for paragraphs**, our primary level of preprocessing is to understand the text by extracting the meaning of the text. This can include sentiments, emotions, intentions, etc. A good way of doing this is to perform abstractive summarization, which involves constructing new text that concisely captures the meaning of the old text.

The actual process of text preprocessing, however, does not necessarily flow distinctly from one level to another. The order of preprocessing steps, as well as whether or not to include specific steps, depends largely on the application. An example process is given here:

1. Segmentation (breaking text into sentences)
2. Spelling correction
3. Noise removal (includes removal of text that would otherwise confuse the main text, including emojis, foreign language, and hyperlinks)
4. Language detection (identifying the language used in a body of text)
5. Stop-words removal (these words are typically high frequency, generic, and less context-specific)
6. Case-folding (removing variances in case such as lowercase, uppercase, titlecase, and so on)
7. Lemmatization
8. Tokenization (break texts into words, phrases, symbols, and other semantically useful units or meaningful elements)
9. Parsing (part of speech identification)
10. Standardization
11. Stemming (reduces tokens to base forms, or **stems**, and removes affixes)

Many of these steps may be unfamiliar - that is perfectly fine, as they will be covered later in this section.

It is commonly accepted that preprocessing text helps improve accuracy, so it is critical to perform good, appropriate preprocessing.

14.2.3 Procedures

Stemming

As mentioned previously, stemming is performed to help normalize our textual data by reducing tokens to their base forms. This is done so that we do not end up with many different words with very similar meaning. For example, stemming turns "watch", "watched", "watching", and "watches" into "watch". To do this, the algorithm accesses a database where known word forms, such as the various forms of "watch", are grouped together. It then analyzes the text for all instances of these forms and turns them all into the root word.

Stemmers also typically perform preprocessing tasks like turning words not at the beginning of a sentence into lowercase and removing **stop words**, which are common words like "the" and "that" which show up in a variety of contexts and typically do not change meaning across these contexts. The `NLTK` library contains a set of English stop words, which the following code prints out. Notice how many of these stop words are common words like pronouns, prepositions, and conjunctions.

```
print(nltk.corpus.stopwords.words('english'))
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you
↪'ve", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him',
↪'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its',
↪'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who
↪', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was',
↪ 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does',
↪'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as',
↪'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between',
↪ 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from',
↪'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',
↪'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any',
↪'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not
↪', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will',
↪'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're',
↪'ve', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't",
↪'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn
↪', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't",
↪'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't",
↪'won', "won't", 'wouldn', "wouldn't"]
```

Stemming algorithms can be very useful for preprocessing text. However, there are always irregularities when using any stemming algorithm. Take the following examples of how a stemming algorithm might transform sentences:

- "I bought an Apple" => *change to lowercase* => "I bought an apple"
  - Here, the stemming algorithm has removed the capital on "Apple". We can clearly see that this drastically altered the meaning; now, the speaker has bought a fruit instead of a product from the company Apple.
- "I watched *The Who*" => *remove stop words* => "I watched"
  - Here, the stemming algorithm has removed the stop words "the" and "who", but this again changed the meaning of the sentence. Instead of watching the TV show *The Who*, the speaker has now watched something unspecified.
- "Take that!" => *remove stop words* => "Take"

- Like the above example, the stemming algorithm removed a stop word, "that". However, the sentence now loses all meaning: it is no longer a complete sentence, and without the "that" which served to add a violent, fighting intent to the original sentence, this emotion and intent is completely gone.

Developed by Martin Porter in 1980, the **Porter Stemmer** is the oldest and most commonly used stemming algorithm in many languages. It is slightly time consuming, so another stemming algorithm that can be used is the **Lancaster Stemmer**. This stemmer is very aggressive, however, and is especially harmful for short tokens, where the token itself may become unclear or altered to a point where it loses meaning.

Let's try these stemmers, as well as a third type, the **Snowball Stemmer**, on some tokens:

```
tokens =  ['narrative', 'comprehensives', 'publicised', 'moderately', 'prosperous',
 ↪'legitimizes', 'according', 'glorifies']

porter = nltk.stem.porter.PorterStemmer() # Porter stemmer
snowball = nltk.stem.snowball.EnglishStemmer() # Snowball stemmer
lancaster = nltk.stem.lancaster.LancasterStemmer() # Lancaster stemmer

# apply each stemmer to the tokens
porter_results = [porter.stem(i) for i in tokens]
snowball_results = [snowball.stem(i) for i in tokens]
lancaster_results = [lancaster.stem(i) for i in tokens]

# display results
pd.DataFrame(
    index = ['Porter Stemmer', 'Snowball Stemmer', 'Lancaster Stemmer'],
    columns = tokens,
    data = [porter_results, snowball_results, lancaster_results]
)
```

```
                  narrative comprehensives publicised moderately prosperous  \
Porter Stemmer         narr     comprehens   publicis      moder    prosper
Snowball Stemmer     narrat     comprehens   publicis      moder    prosper
Lancaster Stemmer       nar     comprehend       publ        mod      prosp

                  legitimizes according glorifies
Porter Stemmer        legitim    accord   glorifi
Snowball Stemmer      legitim    accord   glorifi
Lancaster Stemmer     legitim    accord      glor
```

Exercise 5

How do the different stemmers compare for the list of words in the above example? Which stemmer might have the most trouble on a larger set of tokens?

Exercise 6

Modify the above code to stem your own tokens. How are words commonly shortened, and what types of words seem to cause the most trouble for the stemmers?

Tokenizers

The primary goal of a tokenizer is to break text into tokens. This is highly language dependent; in English, we use white space and punctuation as word and token separators, but this may not work in some languages. Since tokens are semantically useful units of text, a token is not necessarily the same as a word; in fact, some tokens may be pieces of words, or even symbols such as punctuation.

For example, the following code uses `NLTK` to tokenize one of the texts from our UN SDG dataset. Note how not only words like "policy" but also numbers like "70", suffixes like "-'s", and symbols like "%" and "(" have been separated into individual tokens.

```
example = text_df.loc[2425, 'text']
print(f'Original text: "{example}"\n')
print('Tokens:', nltk.word_tokenize(example))
```

```
Original text: "About 70% of them have reported at least one trade policy␣
↪targeting women's economic empowerment. Overall, in four years, almost half of␣
↪the WTO membership has implemented trade policies in support of women (at least␣
↪one). It simply provides examples of trade policies as reported by WTO members␣
↪themselves."

Tokens: ['About', '70', '%', 'of', 'them', 'have', 'reported', 'at', 'least', 'one
↪', 'trade', 'policy', 'targeting', 'women', "'s", 'economic', 'empowerment', '.',
↪ 'Overall', ',', 'in', 'four', 'years', ',', 'almost', 'half', 'of', 'the', 'WTO
↪', 'membership', 'has', 'implemented', 'trade', 'policies', 'in', 'support', 'of
↪', 'women', '(', 'at', 'least', 'one', ')', '.', 'It', 'simply', 'provides',
↪'examples', 'of', 'trade', 'policies', 'as', 'reported', 'by', 'WTO', 'members',
↪'themselves', '.']
```

Exercise 7

Modify the above code to tokenize your own text. Does anything surprise you about how the tokenizer breaks up the text? What makes the resultant tokens semantically useful or meaningful?

For the purposes of building ML models, there are several things that tokenizers track in addition to creating the tokens themselves. Tokenizers track *features*, or the frequency of occurrence of each individual token (normalized or not). They also produce a *sample*, or a vector of all the token frequencies for a given document.

When a tokenizer vectorizes a corpus, it counts the occurrences of tokens in each document, normalizes and weights these occurrences, and constructs an n-gram or other representation of the text, which is discussed in the *next section*.

Named Entity Recognition

The purpose of named entity recognition (NER) is to identify named entities in a text, extract them, and classify them according to their type (people, locations, organizations, and so on). For example, the following code uses the `spaCy` library to recognize named entities in one of the texts from the UN SDG dataset and then displays them with `spaCy`'s built-in `displacy` visualizer.

Note

Before running the following code, you may have to download `en_core_web_sm`. This is a trained pipeline which `spaCy` uses to perform the NLP tasks we've been discussing on English-language texts. You can download it by uncommenting the second line in the code below.

```
## uncomment the next line to download the 'en_core_web_sm' pipeline
# ! python3 -m spacy download en_core_web_sm

nlp = spacy.load('en_core_web_sm')
sample = text_df.loc[5707, 'text']
doc = nlp(sample)
spacy.displacy.render(doc, style='ent', jupyter=True)
```

```
<IPython.core.display.HTML object>
```

At first glance, the meaning of these labels may not be clear. We can use the `spacy.explain()` function to learn more about the meaning of each label:

```
labels = set([ent.label_ for ent in doc.ents])
for label in labels:
    print(f'{label}: {spacy.explain(label)}')
```

```
PERCENT: Percentage, including "%"
GPE: Countries, cities, states
LOC: Non-GPE locations, mountain ranges, bodies of water
DATE: Absolute or relative dates or periods
ORG: Companies, agencies, institutions, etc.
```

In the example above, the trained model has automatically identified and classified various entities in the text. Notice how the types of entities range from countries and organizations to dates and percentages.

Exercise 8

Use the following code to perform NER on a random sample from the UN SDG dataset. Rerun the code several times to get different samples. What types of entities does the model identify?

```
sample = nlp(text_df['text'].sample(1).values[0])
doc = nlp(sample)
spacy.displacy.render(doc, style='ent', jupyter=True)
```

Although the trained model has correctly identified many of the named entities in this text, it has still made some mistakes. For example, it incorrectly labels "Total Primary Energy Demand" as an organization. Additionally, the model often relies on capitalization to identify entities. If we change the sample text to lowercase, notice how the model fails to recognize many of the entities that it previously identified.

```
doc = nlp(sample.lower())
spacy.displacy.render(doc, style='ent', jupyter=True)
```

```
<IPython.core.display.HTML object>
```

Exercise 9

Modify the code from Exercise 8 to perform NER on a random sample from the UN SDG dataset, first in the original case and then in lowercase. (Hint: use the `lower()` function to make the sample lowercase.) Rerun your code several times to get different samples. What mistakes does the model tend to make? What types of entities does it tend to misidentify when the text is converted to lowercase?:::

String Operations

Now that we know about various preprocessing techniques, we can review some basic string operations. These will also be important in identifying various sentences and tokens within our dataset in case we would like to focus on specific words. They can also help in identifying cases, numeric characters, and other tests that we might need to conduct. A few are given below:

```
s.startswith(t) # test if s starts with t
s.endswith(t) # test if s ends with t
t in s # test if t is a substring of s
s.islower() # test if s contains cased characters and all are lowercase
s.isupper() # test if s contains cased characters and all are uppercase
s.isalpha() # test if s is non-empty and all characters in s are alphabetic
s.isalnum() # test if s is non-empty and all characters in s are alphanumeric
s.isdigit() # test if s is non-empty and all characters in s are digits
s.istitle() # test if s contains cased characters and is titlecased (i.e. all words␣
↪in s have initial capitals)
```

For example, the following code uses the `startswith()` function to test if the first text from the UN SDG dataset begins with the string 'The'.

```
text_df.loc[0, 'text'].startswith('The')
```

```
False
```

The functions above are all called on one string at a time. In `pandas`, many of these string operations can also be performed on an entire series or dataframe, not just a single value. (These are known as *vectorized* functions.) For example, the following code uses `str.startswith`, the vectorized version of `startswith`, to test which texts from the UN SDG dataset begin with the string 'The'.

```
text_df['text'].str.startswith('The')
```

```
0        False
1         True
2         True
3         True
4        False
         ...
24664    False
24665    False
24666    False
24667    False
24668     True
Name: text, Length: 24669, dtype: bool
```

Exercise 10

Practice using some of the string operations on your own sample sentences and observe the results.

As an example to work through, let's take the Chinese phrase '□□□□' and see if we can locate it in the texts. This phrase, anglicized as "sigequanmian" and roughly translating to "four comprehensives", was publicized by Chinese President Xi Jinping as a set of goals for China. We will use the vectorized `str.contains` function to search the entire dataset.

```
text_df[text_df['text'].str.contains('□□□□')]
```

```
                                doi                                text_id  \
18009  10.32890/UUMJLS.6.2015.4584  baa468803d1c7dba3c1096b1665ed7f7

                                                    text  sdg  \
18009  In China today,President Xi Jinping's new gran...   16

       labels_negative  labels_positive  agreement
18009                1                4        0.6
```

Another possible string operation is comparison, as given below:

```
t1 = text_df[text_df['text'].str.contains('□□□□')]["text"].values[0]
# compare using '=='
print(t1 == text_df.iloc[18009].text)
# compare casefolded version
print(t1 == text_df.iloc[18009].text.casefold())
# compare lowercase and casefold
print(t1.lower() == text_df.iloc[18009].text.casefold())
```

```
True
False
True
```

We can see that Python uses strict comparisons, meaning that the two sentences will not be equal unless the cases (capital and lowercase) are matched. Note that in the above example, `lower()` and `casefold()` yielded equal results. However, this is not the case for all languages. Take the German letter ß:

```
text = 'groß'

# convert text to lowercase using lower()
print('Using lower():', text.lower())

# convert text to lowercase using casefold()
print('Using casefold():', text.casefold())

# check equality
print(f'{text.lower()} == {text.casefold()}:', text.lower() == text.casefold())
```

```
Using lower(): groß
Using casefold(): gross
groß == gross: False
```

The German word *groß* translates as "large" or "big", and this meaning is kept using `lower()`. However, when using `casefold()`, the word becomes *gross*, which is an acceptable alternate spelling in German but is also used in English to indicate disgust. Even though *groß* and *gross* represent the same word in German, Python's strict comparison will treat them as different. This is another reason to perform consistent and appropriate preprocessing of text.

It is up to you to decide which of these functions to use when converting to lowercase; however, it is best to keep this consistent across all your code so as to avoid potential problems such as this.

14.3 About Text Data

14.3.1 Transformation

ML algorithms of any kind expect numerical features with fixed dimensions. This is especially the case for supervised learning algorithms such as classification algorithms, which is what we will be using to classify our texts into the UN SDG categories.

In documents, however, raw text data is typically semi-structured, with uneven lengths and varied contents. This poses a problem for ML methods, which typically operate on numerical vectors of equal length.

Therefore, once we have our preprocessed text data, our next job is to transform it, and the main way we do this is to *vectorize* it, or transform it into numerical vectors. When we map documents into vectors, we can use several techniques, which we will discuss later:

- bag of words
- n-grams
- word embedding

Vectorization, and transformation of textual data as a whole, depends on tokenization. Carefully chosen preprocessing steps can help reduce complexity and increase ML model accuracy.

Vectorizations

Bag of Words

The **bag of words** approach treats a text or document almost literally as a bag of words; i.e., it ignores the order of the words and simply gathers all the words together in one big collection. All the words are then treated as features, with one feature for each word, where the value of the feature is the number of times that word appears in the document. We then set a fixed number of words to be considered, and our bag of words becomes the vocabulary we work with.

Let's analyze the vocabulary of our UN SDG dataset. We use `CountVectorizer()` from the `scikit-learn` library to create our bag of words, *fitting* it to the corpus to make it learn the vocabulary and *transforming* it into a matrix called `cv_fit` that gives the token counts for each term in each document. (This matrix is called a **document-term matrix**, a concept which will be explained more fully later on in this section.)

```
docs = text_df.text
cv = CountVectorizer()
cv_fit = cv.fit_transform(docs)
```

Each row in `cv_fit` represents a bag of words for the corresponding document. Here are the first five rows:

```
cv_fit[0:5].toarray()
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

We can use the function `get_feature_names_out()` to retrieve the learned vocabulary tokens, which have become the features of the bag of words. For example, the first feature (at index 0) is the token "00", and the last feature (at index 45737) is the token "□□□□".

```
feature_names = cv.get_feature_names_out()
print('Number of features:', len(feature_names))
feature_names
```

```
Number of features: 45738
```

```
array(['00', '000', '0000002', ..., 'œopen', 'ʿadawiyya', '□□□□'],
      dtype=object)
```

We can retrieve the total counts for the features by summing the counts in the matrix. We see, for example, that the token "00" occurs eight times in the entire corpus, while the token "□□□□" occurs twice.

```
total_counts = cv_fit.sum(axis=0)
total_counts
```

```
matrix([[  8, 840,   1, ...,   1,   1,   2]])
```

To get a better idea of how the bag of words works, we can take a look at a short sample text, which is given below.

```
sample_index = 724
print('Sample text:', text_df.text.iloc[sample_index])
sample_bag = cv_fit[sample_index]
```

```
Sample text: Peacemaking, peacekeeping, and peacebuilding may not have the punch␣
↪and the means of national security, but they are receiving an increasing amount␣
↪of attention in education, research, and politics.
```

The `inverse_transform()` function allows us to retrieve a list of all tokens from the vocabulary that occur at least once in the bag of words for this sample text.

```
cv.inverse_transform(sample_bag)[0]
```

```
array(['the', 'of', 'have', 'in', 'and', 'are', 'but', 'education', 'an',
       'research', 'security', 'they', 'may', 'not', 'national',
       'politics', 'amount', 'increasing', 'attention', 'means',
       'peacekeeping', 'receiving', 'peacemaking', 'peacebuilding',
       'punch'], dtype='<U85')
```

The following code compares the counts of several of these tokens in the sample text to their total counts in the entire corpus. We use the attribute `vocabulary_`, which maps the feature names to their indices, to index into the counts.

```
tokens = ['and', 'the', 'peacemaking', 'punch', 'unknown']
pd.DataFrame.from_dict(columns=tokens, orient='index', data={
    'sample count': [sample_bag[0, cv.vocabulary_[token]] for token in tokens],
    'total count': [total_counts[0, cv.vocabulary_[token]] for token in tokens]
})
```

```
                and     the  peacemaking  punch  unknown
sample count      3       2            1      1        0
total count   93357  143100           44      1       39
```

Note how the tokens "and" and "the", which are typically considered to be English stop words, occur more frequently than the other tokens. Meanwhile, the word "unknown" has a sample count of 0 because it does not occur in the sample text, even though it occurs 39 times in the rest of the corpus. We also discover that our sample text contains the only occurrence of the token "punch" in the entire corpus. (Words that occur only once in a corpus are known as *hapax legomena*.)

Exercise 1

What is the most common word in the corpus? (Hint: use `argmax()` on `total_counts` to find the index of the token.)

As we have seen, stop words like "and" and "the" are not removed by default. Fortunately, we can easily modify our code to remove stop words:

```
cv2 = CountVectorizer(stop_words='english') # exclude English stop words
cv2_fit = cv2.fit_transform(text_df.text)
```

`CountVectorizer()` provides a way to easily retrieve the stop words we are removing:

```
print(cv2.get_stop_words())
```

```
frozenset({'describe', 'whither', 'under', 'back', 'than', 'done', 'beside', 'first
 ↪', 'up', 'being', 'moreover', 'he', 'detail', 'whereas', 'get', 'sometimes',
 ↪'herself', 'eleven', 'such', 'my', 'five', 'made', 'sixty', 'three', 'alone',
 ↪'any', 'thus', 'beforehand', 'over', 'which', 'everything', 'eight', 'only',
 ↪'move', 'others', 'therein', 'thereupon', 'amoungst', 'his', 'former', 'meanwhile
 ↪', 'while', 'among', 'for', 'and', 'the', 'own', 'no', 'whence', 'per', 'be',
 ↪'whether', 'too', 'ten', 'behind', 'whole', 'she', 'towards', 'along', 'was',
 ↪'must', 'but', 'once', 're', 'with', 'who', 'by', 'us', 'except', 'some', 'it',
 ↪'have', 'ie', 'ours', 'front', 'whose', 'several', 'indeed', 'rather', 'bill',
 ↪'many', 'further', 'full', 'please', 'formerly', 'into', 'because', 'anywhere',
 ↪'latter', 'via', 'con', 'everywhere', 'etc', 'fire', 'her', 'you', 'becomes',
 ↪'whenever', 'both', 'whereupon', 'beyond', 'from', 'would', 'here', 'thick',
 ↪'anyway', 'in', 'nowhere', 'other', 'where', 'becoming', 'nothing', 'seeming',
 ↪'when', 'somewhere', 'could', 'nine', 'put', 'now', 'cannot', 'also', 'there',
 ↪'nor', 'without', 'below', 'that', 'thin', 'become', 'seem', 'these', 'fifty',
 ↪'they', 'were', 'something', 'can', 'almost', 'anything', 'never', 'show', 'had',
 ↪ 'most', 'inc', 'enough', 'so', 'bottom', 'therefore', 'six', 'hereby', 'four',
 ↪'herein', 'much', 'yourself', 'whereby', 'toward', 'has', 'mill', 'ourselves',
 ↪'de', 'this', 'serious', 'across', 'system', 'couldnt', 'what', 'always', 'take',
 ↪ 'nevertheless', 'last', 'very', 'seems', 'twenty', 'against', 'whoever', 'am',
 ↪'less', 'fifteen', 'hers', 'out', 'after', 'afterwards', 'down', 'few', 'well',
 ↪'thru', 'not', 'sincere', 'mine', 'me', 'between', 'do', 'interest', 'third',
 ↪'ever', 'each', 'side', 'before', 'an', 'during', 'anyhow', 'around', 'as',
 ↪'forty', 'hereupon', 'hence', 'another', 'two', 'eg', 'at', 'may', 'hundred', 'of
 ↪', 'one', 'every', 'those', 'are', 'hereafter', 'find', 'someone', 'sometime',
 ↪'to', 'found', 'why', 'though', 'him', 'otherwise', 'yours', 'keep', 'our', 'will
 ↪', 'still', 'wherever', 'co', 'latterly', 'onto', 'everyone', 'themselves',
 ↪'about', 'throughout', 'became', 'through', 'although', 'cant', 'within', 'might
 ↪', 'yet', 'perhaps', 'top', 'hasnt', 'whom', 'call', 'already', 'part', 'however
 ↪', 'again', 'twelve', 'thereafter', 'either', 'wherein', 'thereby', 'else', 'them
 ↪', 'thence', 'how', 'if', 'upon', 'its', 'since', 'whereafter', 'amongst', 'empty
 ↪', 'mostly', 'name', 'together', 'i', 'should', 'we', 'or', 'is', 'fill',
 ↪'neither', 'noone', 'until', 'cry', 'besides', 'itself', 'somehow', 'whatever',
 ↪'all', 'due', 'their', 'a', 'least', 'next', 'your', 'elsewhere', 'often', 'then
 ↪', 'been', 'see', 'seemed', 'un', 'more', 'go', 'yourselves', 'even', 'off',
 ↪'himself', 'myself', 'amount', 'give', 'none', 'same', 'ltd', 'above', 'namely',
 ↪'anyone', 'on', 'nobody'})
```

Exercise 2

Find the length of the vocabulary (the number of features) without stop words, and also find the length of the list of stop words. Is the difference between our original vocabulary and our vocabulary with stop words removed equal to the number of stop words? Why or why not?

Exercise 3

What is the most common word now that stop words have been removed?

Now that stop words have been excluded from the vocabulary, the list of vocabulary tokens in our sample text with non-zero counts has changed as well.

```
sample_bag = cv2_fit[sample_index]
cv2.inverse_transform(sample_bag)[0]
```

```
array(['education', 'research', 'security', 'national', 'politics',
       'increasing', 'attention', 'means', 'peacekeeping', 'receiving',
       'peacemaking', 'peacebuilding', 'punch'], dtype='<U85')
```

One-Hot Vectors

Another vectorization approach is to treat a text as a **one-hot vector**. This approach lines up all words in the vocabulary and assigns each of them an index $1 \leq i \leq V$, where V is the number of words in the vocabulary. For each word, it then forms a vector

$\vec{w} = (0, 0, ..., 0, 1, 0, 0, ..., 0)$,

with only one 1 in the vector and the rest of the numbers being zeroes. The 1 is at the position i that indexes the word in our vocabulary. The order of the words is ignored, so it does not matter if a certain word is indexed before another one. Because all of the words in our vocabulary are unique, there are no similarity notions among the vectors.

To gain a better understanding of how one-hot vectors work, we can take a look at how a short sample text from the UN SDG dataset would be encoded. As before, we use `NLTK` to tokenize the text.

```
sample_text = text_df.text.iloc[12737].lower()
print('Sample text:', sample_text)
tokens = nltk.word_tokenize(sample_text)
print('Tokens:', tokens)
```

```
Sample text: the chapter examines the contribution that international relations␣
↪theory has made to the reading and practice of peacebuilding.
Tokens: ['the', 'chapter', 'examines', 'the', 'contribution', 'that',
↪'international', 'relations', 'theory', 'has', 'made', 'to', 'the', 'reading',
↪'and', 'practice', 'of', 'peacebuilding', '.']
```

We also define a convenience function to reshape the list of tokens into a format that can be used by the one-hot encoder:

```
def ohe_reshape(tokens):
    return np.asarray(tokens).reshape(-1,1)
```

We use `OneHotEncoder()` from the `scikit-learn` library to create the one-hot encoding. We fit the encoder to our sample text to make it learn the vocabulary. As with `CountVectorizer`, we can view the resulting features using the function `get_feature_names_out()`.

```
ohe = OneHotEncoder(handle_unknown='ignore') # encode unknown tokens as vectors of␣
↪all zeros
ohe.fit(ohe_reshape(tokens))
ohe.get_feature_names_out()
```

```
array(['x0_.', 'x0_and', 'x0_chapter', 'x0_contribution', 'x0_examines',
       'x0_has', 'x0_international', 'x0_made', 'x0_of',
       'x0_peacebuilding', 'x0_practice', 'x0_reading', 'x0_relations',
       'x0_that', 'x0_the', 'x0_theory', 'x0_to'], dtype=object)
```

Now that the encoder has learned the vocabulary, we can try it out on text. The following code transforms the first five tokens of our sample text into one-hot encodings by calling the `transform` function:

```
print('First five tokens:', tokens[:5])
ohe.transform(ohe_reshape(tokens[:5])).toarray()
```

```
First five tokens: ['the', 'chapter', 'examines', 'the', 'contribution']
```

```
array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.,
        0.],
       [0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.,
        0.],
       [0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0.]])
```

Note how, as explained above, each unique token in the learned vocabulary has a unique encoding, with only one 1 in the vector. Note also how the first and fourth vectors in this matrix are identical because they represent the same token, "the".

We can view the unique encoding of "the" by using the `transform` function again:

```
ohe.transform(ohe_reshape(['the'])).toarray()[0]
```

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.])
```

If we try to encode a token that isn't included in the learned vocabulary, however, we get a vector of only zeros:

```
ohe.transform(ohe_reshape(['unknown'])).toarray()[0]
```

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

Exercise 4

Verify that each token in the learned vocabulary has a unique encoding.

Numeric Count Vectors

A third vectorization approach is to make a numeric vector of counts. This approach counts the occurrence of the words in the document and lines up the counts of words in a row for each document. Numeric count vectors are similar to document-term matrices, although they differ in that they measure words rather than tokens (which can be words, but can also be larger units such as phrases or smaller units such as affixes). As a result, we will focus on document-term matrices, which are covered in the next part of this section.

Document-Term Matrices

One final vectorization approach we will introduce here is to form a document-term matrix. To form this matrix, we define the matrix's columns to be tokens (such as words or phrases), the matrix's rows to be the documents in the collection, and the value of each entry to be the frequency of each token occurring in the document. Additionally, the values in each entry are weighted frequencies. One weighting method we use is **TF-IDF**, or term frequency (TF) times inverse document frequency (IDF). TF refers to the frequency of the word in the document, while IDF refers to the inverse of the number of documents containing the word divided by number of documents. TF-IDF is discussed later in this section.

Below, we compute a simple document-term matrix for a portion of the documents and tokens in the UN SDG dataset. The features are all unigrams (one-word tokens). While the unigram does provide context for each word, NLP tasks built on such simple model would be disadvantaged.

```
docs = text_df.text
count_vectorizer = CountVectorizer(stop_words='english')
count_vector = count_vectorizer.fit_transform(docs).toarray()
count_vector_df = pd.DataFrame(count_vector, columns=count_vectorizer.get_feature_
↪names_out())
term_freq = pd.DataFrame({"term": count_vector_df.columns.values, "freq" : count_
↪vector_df.sum(axis=0)})
count_vector_df.loc[110:120,term_freq.sort_values(by="freq", ascending=False)[:10].
↪term] # take a portion
```

	countries	women	development	health	water	public	social	education	\
110	0	0	0	0	0	0	0	0	
111	0	0	0	0	0	1	0	0	
112	0	0	4	0	0	0	0	0	
113	0	0	0	1	0	0	0	0	
114	0	0	0	1	0	0	0	0	
115	0	0	0	0	0	0	0	0	
116	0	0	2	0	0	0	0	0	
117	2	0	0	0	0	0	1	0	
118	0	4	4	1	0	0	1	0	
119	0	0	0	0	0	0	0	0	
120	0	2	0	0	0	1	0	0	

	policy	international
110	0	0
111	0	3
112	0	1
113	0	0
114	0	0
115	0	0
116	1	0
117	0	0
118	0	0

(continues on next page)

(continued from previous page)

```
119        0                0
120        0                0
```

Exercise 5

Look at document 118.

- What are the most frequent words for this document?
- Refer to the UN SDGs. What is your best guess as to which goal this document describes?

N-Grams

An n-gram is a contiguous sequence of n items from a text. Items in an n-gram are tokens we choose to include through preprocessing, but these items can also include markers we add to the data to serve the NLP tasks at hand.

For example, if we were to find what words are often used at the beginning or end of a sentence, we would add <s> and </s> as markers for the beginnings and ends of sentences, to denote the sentence boundaries. We would then build bigrams ($n = 2$); for the sentence "I built an AI machine", we would get the items (<s> I), (I built), (built an), (an AI), (AI machine), and (machine </s>). We then count the frequency of bigrams. specifically of the form (<s> *) and (* </s>), where * represents any other token.

In the previous subsection, we looked at a document-term matrix of unigrams (with stop words removed). We can check the total count of unigrams — as well as the number of *unique* unigrams — with the following code. Notice how the second number is smaller than the first because many of the unigrams occur more than once in the corpus.

```
print('Total count of unigrams (without stop words):', count_vector.sum())
print('Number of unique unigrams (without stop words):', len(count_vectorizer.
 ↪vocabulary_))
```

```
Total count of unigrams (without stop words): 1351051
Number of unique unigrams (without stop words): 45440
```

We then compare this to the counts obtained when stop words are not removed, which we calculate using the `cv` and `cv_fit` objects from earlier in this section:

```
print('Total count of unigrams (with stop words):', cv_fit.sum())
print('Number of unique unigrams (with stop words):', len(cv.vocabulary_))
```

```
Total count of unigrams (with stop words): 2335467
Number of unique unigrams (with stop words): 45738
```

Notice that stop word removal alone reduced the total count of terms by around 42%, even though it decreased the number of unique terms in the vocabulary only slightly.

Next, we can check the bigrams for our document corpus by setting the `ngram_range` parameter to `CountVectorizer`:

```
count_vectorizer = CountVectorizer(ngram_range=(2,2), stop_words='english')
count_vector = count_vectorizer.fit_transform(docs)
print('Total count of bigrams (without stop words):', count_vector.sum())
print('Number of unique bigrams (without stop words):', len(count_vectorizer.
 ↪vocabulary_))
```

```
Total count of bigrams (without stop words): 1326382
Number of unique bigrams (without stop words): 834465
```

Exercise 6

Notice that the number of unique bigrams is larger than the number of unique unigrams, even though the total count of bigrams is smaller than the total count of unigrams. Why might this be the case?

Exercise 7

Modify the above code to get the counts of the trigrams without stop words. Is the total count of trigrams larger or smaller than the total count of bigrams? What about the number of unique trigrams compared to the number of unique bigrams? Why might this be the case?

Probabilistic Language Modeling

So far, we have worked with clearly defined texts and sequences of words. But sometimes, the texts we get are incomplete, or we might want to reduce computational time by only giving portions of the text. As such, we might turn to probabilistic language modeling. Typically, a probabilistic language model analyzes a body of text data and computes the following based on that body of text data:

- the probability of a sentence or sequence of words occurring, $P(w_1, w_2, w_3, ..., w_n)$,
- the probability of the next word given k words, $P(w_n|w_{(n-k)}, w_{(n-(k-1))}, ..., w_{(n-1)})$.

We then use the chain rule to compute the joint probability:

$P(A \cap B) = P(B|A) * P(A)$

$\implies P(w_1, w_2, w_3, ..., w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)...P(w_n|w_1, ..., w_{(n-1)})$.

We could estimate the probability by calculating

$\text{count}(w_1, w_2, ..., w_{(i-1)}, w_i)/\text{count}(w_1, w_2, ..., w_{(i-1)})$,

but with too many possible occurrences, this calculation is not feasible.

Instead, we can simplify the equation through the Markov Assumption, which in an NLP context assumes that the next word only depends on some previous words, not all of them. This means we can place a condition on the previous k words rather than all the previous words:

$P(w_n|w_1, w_2, w_3, ..., w_{(n-1)}) \sim P(w_n|w_{(n-k)}, ..., w_{(n-1)})$.

When considering a unigram $(n = 1)$ model, we calculate the probability as

$P(w_1, w_2, w_3, ..., w_n) = P(w_1)P(w_2)P(w_3)...P(w_n)$.

A similar process occurs when considering a bigram $(n = 2)$ model, where we place a condition on the previous word,

$P(w_n|w_1, ..., w_{(n-1)}) \sim P(w_i|w_{(i-1)})$,

and we can similarly extend this process to trigrams, 4-grams, 5-grams, or any n-gram.

However, no matter how many n we consider, we still will not arrive at a sufficient model; language has long-distance dependencies that require extremely large n that can even extend to previous sentences and/or documents. To truly determine if our n-gram is sufficient, we extrinsically evaluate it by putting our model to tasks.

For example, if we only want to deal with spell correction, a unigram be sufficient, but for machine translation, we might need to work with very long sentences. Google's recent speech model features 20B sentences (https://arxiv.org/abs/2303.01037?utm_source=substack&utm_medium=email).

Natural Language Generation

Natural Language Generation (NLG) is an offshoot of Probabilistic Language Modeling which is used to generate texts. Given a prompt (such as a short incomplete sentence), NLG models will generate text in response (such as giving the next few words of the sentence).

Exercise 8

Visit this Hugging Face demo, which attempts to predict the next token when given part of a sentence: https://alonsosilva-nexttokenprediction.hf.space/. Try typing in a few incomplete sentences. How accurate are the model's predictions?

As the exercise shows, these language models are not perfect. But especially with recent developments in AI, they are improving.

Another main role of NLG is automatic summarization of a text. The NLG has the ability to perform extractive summarization (extracting meaning from a text), or abstractive summarization (constructing an abstract from a text). NLG can also be used for topic modeling to identify the main topics of the text, or as an advanced machine translation tool.

An example of an advanced natural language generator is the Generative Pre-trained Transformer (GPT), mainly pioneered by the startup OpenAI. The GPT is a text generation deep learning model trained on massive datasets, including internet data, book data, GitHub data, and more. GPTs, including the famous ChatGPT, can take thousands of words as input and have been trained on billions of different parameters.

14.3.2 Features

TF-IDF (Term Frequency, Inverse Document Frequency)

In the document-term matrix previously discussed, we represented columns as tokens, rows as documents, and the entry values as the number of occurrences of that token in the document. However, using the counts alone may not give us an accurate picture of how significant each token is in context, since some tokens naturally occur more frequently than others. We can ameliorate this problem by weighting the frequencies with TF-IDF.

In our computation of the matrix cell (d, t) with document as row and term as column, we compute the following:

- TF: term frequency of term t in the document d = count of term t in document d divided by total number of terms in document d
- DF: document frequency of term t = number of documents containing term t divided by total number of documents $\frac{\text{df}}{N}$, where N is the total number of documents and is a constant that can be ignored
- IDF: take the inverse of DF
- TF-IDF = TF * IDF

We can use `scikit-learn` to get TF-IDF with `scikit-learn.feature_extraction.text.TfidfVectorizer`. This code takes several arguments:

- `sublinear_df` : Boolean; if true, uses a logarithmic form for frequency,
- `min_df` : the minimum number of documents a word must be present in to be kept,
- `norm`: usually set to l2 to ensure all our feature vectors have a Euclidean norm of 1,

- `ngram_range`: determines whether to take unigrams, bigrams, or both, and
- `stop_words`: usually set to "english" for removing English stop words.

Note that when using logarithmic form for term frequency, the weighted TF is transformed into $1 + \log \text{TF}$; if $\text{TF} > 0$, then $\text{TF} = 0$; otherwise, we use $\log \frac{N}{\text{df}}$ for IDF to dampen the effect of $\frac{N}{\text{df}}$.

As an example, we compute the same document-term matrix as before, but this time using `TfidfVectorizer`. We include only unigrams and remove English stop words.

```
docs = text_df.text
tfidf_vectorizer = TfidfVectorizer(ngram_range=(1,1), stop_words='english')
tfidf_vector = tfidf_vectorizer.fit_transform(docs).toarray()
tfidf_vector_df = pd.DataFrame(tfidf_vector, columns=tfidf_vectorizer.get_feature_
↪names_out())
term_freq = pd.DataFrame({"term": tfidf_vector_df.columns.values, "freq" : tfidf_
↪vector_df.sum(axis=0)})
tfidf_vector_df.loc[110:120,term_freq.sort_values(by="freq", ascending=False)[:10].
↪term] # take a portion
```

```
     countries     women  water    health  development  education    energy  \
110   0.000000  0.000000    0.0  0.000000     0.000000        0.0  0.144528
111   0.000000  0.000000    0.0  0.000000     0.000000        0.0  0.000000
112   0.000000  0.000000    0.0  0.000000     0.150946        0.0  0.000000
113   0.000000  0.000000    0.0  0.073888     0.000000        0.0  0.000000
114   0.000000  0.000000    0.0  0.105940     0.000000        0.0  0.000000
115   0.000000  0.000000    0.0  0.000000     0.000000        0.0  0.060643
116   0.000000  0.000000    0.0  0.000000     0.128756        0.0  0.000000
117   0.134099  0.000000    0.0  0.000000     0.000000        0.0  0.000000
118   0.000000  0.244198    0.0  0.063116     0.216640        0.0  0.000000
119   0.000000  0.000000    0.0  0.000000     0.000000        0.0  0.000000
120   0.000000  0.097945    0.0  0.000000     0.000000        0.0  0.000000

       social    public   policy
110  0.000000  0.000000  0.00000
111  0.000000  0.055794  0.00000
112  0.000000  0.000000  0.00000
113  0.000000  0.000000  0.00000
114  0.000000  0.000000  0.00000
115  0.000000  0.000000  0.00000
116  0.000000  0.000000  0.06901
117  0.082325  0.000000  0.00000
118  0.059071  0.000000  0.00000
119  0.000000  0.000000  0.00000
120  0.000000  0.046368  0.00000
```

Feature Effectiveness

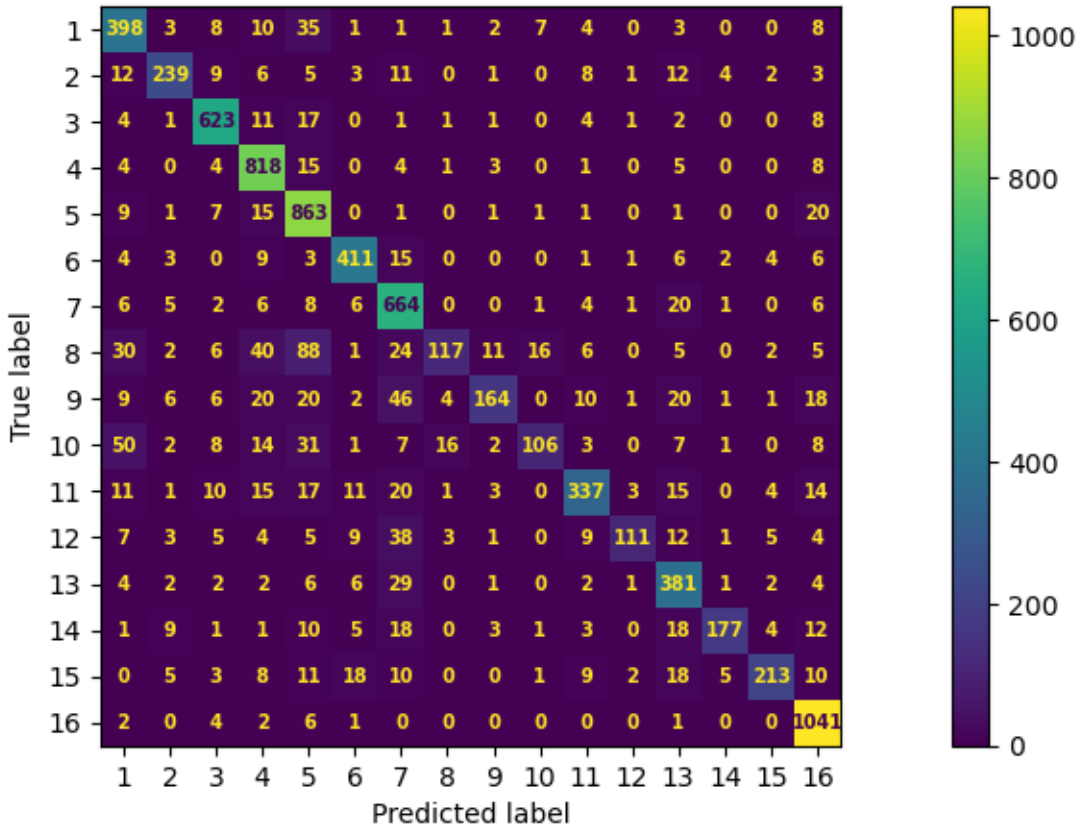
In examining how to evaluate the effectiveness of features in the context of NLP tasks, we will mainly use the task of document classification and classifying our texts into SDG categories as mentioned in the preface of this section.

When looking at document classification, we can use a confusion matrix and get tp, fn, fp, tn; these can be used to calculate the precision, recall, and f1 measures as follows:

- precision: $\frac{tp}{tp+fp}$,
- recall: $\frac{tp}{tp+fn}$,
- f1: $2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$.

Below, we use a multinomial Naive Bayes algorithm to classify our documents by their SDG; the process of this is explained in later sections. For now, we examine the multiclass confusion matrix classifying documents by their respective UN SDG:

```
docs = text_df.text
categories = text_df.sdg
X_train, X_test, y_train, y_test = \
    train_test_split(docs, categories, test_size=0.33, random_state=7)

X_train_count_vectorizer = CountVectorizer(ngram_range=(2,2), stop_words = "english" )
X_train_count_vectorizer.fit(X_train)
X_train_count_vector = X_train_count_vectorizer.transform(X_train)
X_test_count_vector = X_train_count_vectorizer.transform(X_test)

count_multinomialNB_clf = MultinomialNB().fit(X_train_count_vector, y_train)
y_pred = count_multinomialNB_clf.predict(X_test_count_vector)
fig, ax = plt.subplots(figsize=(15, 5))
font = {'family': 'sans-serif', 'weight': 'heavy','size': 7,}
ConfusionMatrixDisplay.from_predictions(y_test, y_pred, text_kw=font, ax=ax, cmap=mpl.
↪colormaps["viridis"]);
```



When looking at the metrics for SDG 1, we find that:

- $tp = 398$ (where predicted label and true label are the same),
- $fp = 153$ (calculated by summing vertically below 398),
- $fn = 83$ (calculated by summing horizontally to the right of 398).

Exercise 9

Use the above values and formulas to calculate precision, recall, and f1 for SDG 1.

14.3.3 More Exercises

Exercise 10

Using the UN SDG dataset, begin working with some of the `scikit-learn` NLP capabilities, such as the following code:

```
docs = text_df.text
count_vectorizer = CountVectorizer()
count_vector = count_vectorizer.fit_transform(docs).toarray()
count_vector_df = pd.DataFrame(count_vector, columns=count_vectorizer.get_feature_
↪names_out())
```

Try out different parameters in `CountVectorizer`, including

```
ngram_range = (1,2)
stop_words='english'
min_df = 2
min_df = 3
```

What are the most frequent unigrams, bigrams, and trigrams? Answer this question with, and then without, stop word removal.

Exercise 11

Write a function that takes in a document corpus like the UN SDG dataset and returns:

- The top 50 most frequent words,
- A plot of the cumulative word count from the most frequent word to the 50th most frequent, and
- A comparison of the level of cumulation (i.e., the height where the plot ends) with the total number of words of the input corpus, outputting the percentage.

Your function should also contain a parameter `stop_words` with its default set to `None`, which does not remove stop words.

Exercise 12

Run your function from Exercise 11 on the documents labeled with SDG 8. What is the output?

Exercise 13

Run your function from Exercise 11 on the entire UN SDG corpus, once with stop removal and once without. Describe the similarities and differences in the outputs when this parameter is changed.

14.4 Document Embedding

In the previous section, we represented variable length texts as fixed length numeric vectors; the approach we have used so far is the traditional approach of Bag of Words (BoW), which tokenizes a text into words (tokens). BoW ignores the order of the tokens, though it may pay attention to the frequency. This approach is high dimension, and very sparse; this may result in overfitting and high time complexity.

A more modern text vectorization approach is **word embedding** (also called simply embedding), relying on neural representations. This approach takes distributional semantics into account; that is, a word's meaning is inferred from the words that frequently appear close-by. Hence, we can construct a word's context by using the set of words that appear nearby within a fixed-sized window.

Semantically similar texts, then, would appear closer to each other in the vector space. We could also possibly capture semantic relationships by operations in the vector space; for example, similarity between texts could be measured by vector dot product. We could also perform algebraic operations; for example,

vector(``King'') − vector(``Man'') + vector(``Woman'') ∼ vector(``Queen'').

Modern-day representations are typically learned from vast body of texts, often with deep neural networks, and they typically result in pre-trained models.

To get embeddings, we use the `tensorflow` library, installed and imported as follows:

```
## uncomment the next two lines of code to install tensorflow
# ! pip install tensorflow
# ! pip install tensorflow_hub

import tensorflow as tf
import tensorflow_hub as hub
```

Warning

The `tensorflow` package can be finicky, especially on certain distributions of Python and on computers that are not very powerful. However, it is a versatile and useful package for document embedding, which is an important part of natural language processing. If you encounter trouble with `tensorflow`, try using a kernel on a different version of Python (such as Python 3.9) and/or a reduced version of the dataset. If necessary, the code from this section can be skipped, and later sections will not depend on code from this one.

To start, we first need to break our document corpus into sentences, using a tokenizer from `nltk`:

```
def tokenize_into_sentences(corpus):
    corpus_sentence = []
    corpus_sdg = []
    corpus_sample = []
    for (text, sdg, i) in iter(zip(corpus.text, corpus.sdg, corpus.index)):
        sentences = nltk.sent_tokenize(text)
        corpus_sentence += sentences
        corpus_sdg += [sdg]*len(sentences)
        corpus_sample += [i]*len(sentences)
    sentence_df = pd.DataFrame({"text": corpus_sentence, "sdg": corpus_sdg, "sample":␣
↪corpus_sample})
    return sentence_df

sentence_df = tokenize_into_sentences(text_df)
```

The dataframe `sentence_df` now contains all of the sentences in the corpus. The column `text` gives the sentences,

the column `sdg` gives the SDG, and the column `sample` gives the index of the original text sample from which the sentence was extracted. We can use the `head` function to see the first few sentences:

```
sentence_df.head(10)
```

```
                                                text  sdg  sample
0  "From a gender perspective, Paulgaard points o...    5       0
1  But the fact that young people are still worki...    5       0
2  When Paulgaard refers to continuity with tradi...    5       0
3  As described earlier, Paulgaard (2015) conclud...    5       0
4  The average figure also masks large difference...    3       1
5  The number of annual contacts ranges from 2.0 ...    3       1
6  In addition, poor coverage of outpatient presc...    3       1
7  These findings are consistent with previous wo...   10       2
8  Returns to education are also found to be an i...   10       2
9  In these countries, however, the magnitude of ...   10       2
```

Exercise 1

What are the dimensions of the `text_df` and `sentence_df` dataframes? What do each of the dimension numbers represent?

Exercise 2

Evaluate the performance of the sentence tokenizer by picking one of the sample texts, manually breaking it into sentences, and determining whether your sentence divisions match the corresponding ones in `sentence_df`.

An important question to ask is how many sentences each document has; in other words, what is the distribution of the number of sentences in each text? We can determine that by *grouping* the dataframe by sample, counting the number of sentences in each group, and then counting those counts:

```
sentence_df.groupby(by='sample').count()['text'].value_counts()
```

```
text
3     12015
4      7579
5      2263
6      1586
2       586
7       354
8       119
1        89
9        21
10       14
12       12
13        6
15        6
11        4
14        4
16        3
17        2
18        1
24        1
```

(continues on next page)

(continued from previous page)

```
31        1
21        1
19        1
20        1
Name: count, dtype: int64
```

Notice that the vast majority of the documents have less than 10 sentences, with most being only 3 or 4 sentences long.

This type of sentence tokenization can help us with further NLP tasks down the line. We can now turn to the embedding method itself.

14.4.1 Universal Sentence Encoder

The Universal Sentence Encoder (USE) was first published by Google around 2018. It maps a sentence, word, or short paragraph to a fixed length (typically 512) numeric vector. This approach would mean semantically similar sentences would be placed closer to each other in the embedding space.

Embeddings are typically the result of using raw text, so no pre-processing would be involved. This sentence embedding can then be used for downstream applications such as classification, clustering, and language prediction.

USE is a pre-trained model trained on variety of data such as books and Wikipedia. It was trained with a deep averaging network (DAN) encoder; more information about the process behind USE can be found at https://arxiv.org/pdf/1803.11175.pdf.

To utilize USE, we can take one of three approaches:

1. We could take our desired document, turn it into a collection of sentences, and then map each sentence to its respective vector;
2. We could treat each document as a short paragraph and match each document to its respective vector, or;
3. We could take a similar approach to #1, except then aggregate the vectors for each document to form a single vector per document.

USE can be found at https://tfhub.dev/google/universal-sentence-encoder/4. Go to this link, download the embedding (which is around 1GB in size), and put it in a directory you can access. Change the next line of code to refer to this directory.

```
# change this to your own embedding directory
embedding_dir = "embeddings/"

# load the embedding
embed = hub.load(embedding_dir + "universal-sentence-encoder_4")
```

Note that the first time you run this, it may take some time (5+ minutes) to complete the process.

We are now ready to use USE! We will be returning back to embedding in more depth in Section 6, but as an example, we will embed some training and testing vectors:

```
docs = sentence_df.text
categories = sentence_df.sdg
X_train, X_test, y_train, y_test = \
    train_test_split(docs, categories, test_size=0.33, random_state=7)

X_train_use_vector = embed(X_train.tolist())
X_test_use_vector = embed(X_test.tolist())
```

Here is what the embedding of the training vector looks like:

```
X_train_use_vector
```

```
<tf.Tensor: shape=(62202, 512), dtype=float32, numpy=
array([[ 0.05109644,  0.05008151, -0.00049568, ..., -0.02072422,
         0.03613124,  0.01162022],
       [ 0.06358109, -0.0565166 , -0.05213946, ...,  0.00228676,
        -0.04131475, -0.00125164],
       [ 0.05327134, -0.02345247,  0.02675822, ...,  0.06473472,
         0.0628962 ,  0.03613978],
       ...,
       [ 0.01227464,  0.00510351, -0.00631195, ...,  0.0456263 ,
        -0.04639212, -0.07870013],
       [ 0.06549167,  0.08835391,  0.00940975, ...,  0.03276522,
         0.02704752, -0.00052535],
       [-0.04562642, -0.00093942, -0.0363333 , ..., -0.03696235,
        -0.03747366, -0.03414274]], dtype=float32)>
```

And here is what the embedding of the testing vector looks like:

```
X_test_use_vector
```

```
<tf.Tensor: shape=(30637, 512), dtype=float32, numpy=
array([[-0.06285849,  0.05835171, -0.0444552 , ..., -0.02508012,
        -0.04616681, -0.02821783],
       [ 0.04709839, -0.03887603,  0.01155359, ...,  0.02760062,
        -0.06390072, -0.0412748 ],
       [ 0.01141804, -0.01593475, -0.04189894, ...,  0.05544081,
         0.03996534,  0.03494091],
       ...,
       [ 0.07271196, -0.01527977, -0.02277797, ...,  0.02758262,
        -0.00676865,  0.05427777],
       [-0.07911933, -0.03813467,  0.00067008, ...,  0.00562062,
         0.06976797, -0.041276  ],
       [-0.04857083,  0.06084292, -0.03894068, ...,  0.01013256,
        -0.03109165, -0.07248887]], dtype=float32)>
```

Exercise 3

Take two documents, one labeled as SDG 1 and the other as SDG 8. Segment these into sentences, compute the embeddings, and find the dot product between the embeddings.

```
# import libraries
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
import matplotlib as mpl
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.neural_network import MLPClassifier

from sklearn.ensemble import RandomForestClassifier
```

(continues on next page)

(continued from previous page)

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

# change this to your own data directory
data_dir = "data/"

# read and preprocess data
text_file_name = "osdg-community-data-v2023-01-01.csv"
text_df = pd.read_csv(data_dir + text_file_name,sep = "\t",  quotechar='"')
col_names = text_df.columns.values[0].split('\t')
text_df[col_names] = text_df[text_df.columns.values[0]].apply(lambda x: pd.
 ↪Series(str(x).split("\t")))
text_df = text_df.astype({'sdg':int, 'labels_negative': int, 'labels_positive':int,
 ↪'agreement': float}, copy=True)
text_df.drop(text_df.columns.values[0], axis=1, inplace=True)
text_df = text_df.query("agreement > 0.5 and (labels_positive - labels_negative) > 2")
text_df.reset_index(inplace=True, drop=True)
```

14.5 Classification of Texts

Now that we have gone through the arduous process of preprocessing, transforming, and embedding our text data, we can finally get into some of the ways we can work with it, most prominent of which is classification. In the case of our UN SDG dataset, our goal is to classify documents by the goal they match with. In this subsection, we will briefly review the general process of machine learning before going into some of the methods by which text data are commonly classified.

14.5.1 A Machine Learning Overview

Machine learning of any kind can be boiled down to the process by which we train a machine to make the most effective predictions. When we train a machine to make numerical predictions, we perform **regression**; if we train a machine to categorize data into categories, we perform **classification**. Since our UN SDG dataset is classified into the categories of the goals, we will focus on classification.

There are two main types of machine learning: **supervised** and **unsupervised**. Supervised learning means that the data we give the machine is already labeled with the "correct" value or category, while unsupervised learning means that the data is unlabeled and there is no "correct" value or category. Here, since our UN SDG data is already labeled with the goal it aligns most with, we consider it supervised. As such, the goal for the machine is to match the "correct" labels of the data we give it.

Machine learning starts with a dataset; with modern advances in data collection and storage, machine learning datasets typically involve thousands of data points, while some industry machine learning projects have access to more powerful computers and can use datasets with millions of data points. This dataset is split into a **training set** and a **testing set**. The training set consists of the data fed to the machine for it to build its model, while the testing set consists of data kept separate to evaluate the model's performance. Typically, the training set consists of more data than the testing set; it is common to split the main set 70/30 (70% training, 30% testing) or 80/20.

When a machine builds it predictive model, we need to understand that no machine is perfect at predicting (though some can get pretty close). As such, the main goal of an algorithm when given the training set is to **minimize error**. The process of error minimization varies between algorithms; one common process is gradient descent, which aims to find a minimum for the error function (a function which determines the error based on the function's internal parameters).

However, to truly judge the efficacy of the algorithm, we must use data that the algorithm has not yet seen. Hence, the testing set is used to evaluate the algorithm's performance on data similar (but different) from the training set. As data

scientists, our goal would be to maximize the performance on the testing set, and we do so by several means discussed in section 6.

For working with all the algorithms, we will use the same train/test split of 66/33:

```
docs = text_df.text
categories = text_df.sdg
X_train, X_test, y_train, y_test = \
    train_test_split(docs, categories, test_size=0.33, random_state=7)
```

We then vectorize using TF-IDF from the previous section; for this, we are using bi-grams. Unigrams will be given as exercises throughout the section.

```
X_train_tfidf_vectorizer = TfidfVectorizer(ngram_range=(2,2), stop_words = "english",␣
↪min_df=5)
X_train_tfidf_vectorizer.fit(X_train)
X_train_tfidf_vector = X_train_tfidf_vectorizer.transform(X_train)
X_test_tfidf_vector = X_train_tfidf_vectorizer.transform(X_test)
```

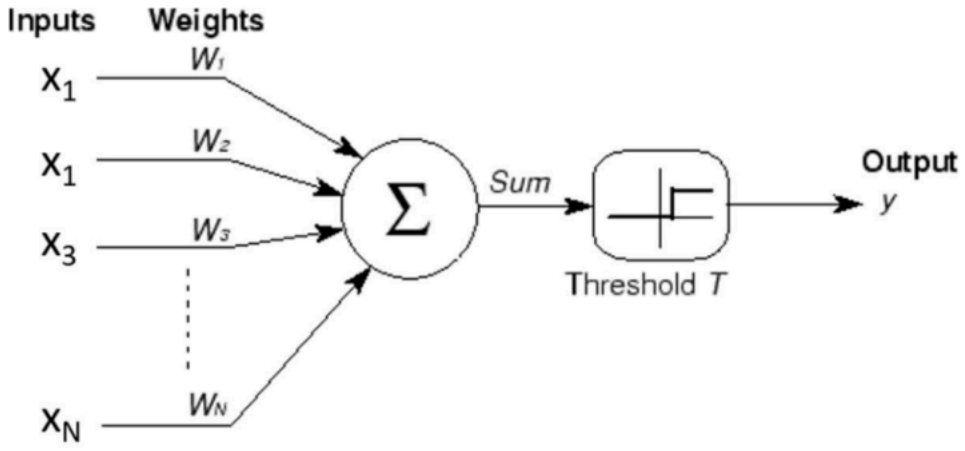
14.5.2 The Algorithms

Many algorithms exist for machine learning predictions. Some, like the Support Vector Machine and Linear/Multiple Regression, have existed for a long time and are applicable to a wide range of problems, while others are more recent and are meant to be high-performance on more specific problems. Here, we present three algorithms most commonly used to classify text data.

The Perceptron

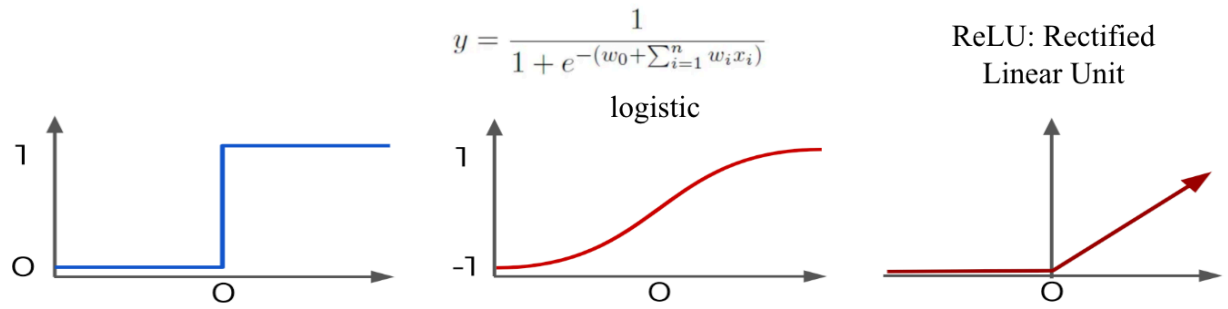
In 1943, McCulloch & Pitts developed the first Perceptron, called the Neuron Model. This model was first used in 1958 by Frank Rosenblatt, who formed the classical perceptron model, and in 1969, Minsky and Papert published extensive research on the perceptron model.

The single perceptron is a binary classifier and takes each input x_i and multiplies it by some corresponding weight w_i. These are then added for all x and w, or $\sum_{i=1}^{N} x_i w_i$. This sum is compared to a threshold T, producing an output of 0 if below T or 1 if above T.



https://news.cornell.edu/stories/2019/09/professors-perceptron-paved-way-ai-60-years-too-soon

This kind of function is called an activation function, which generally involve a value of 0 or -1 until a threshold O is reached, after which the function stays at 1 or increases. Alternative activation functions include logistic regression (below middle) or the rectified linear unit (ReLU) (below right), with the perceptron at bottom left for reference.



It was proven relatively quickly that in addition to only being a binary classifier, the perceptron can only handle linearly separable data. These problems can be fixed by using a multilayer perceptron.

Multilayer perceptrons combine several perceptrons in at least one hidden layer, each layer feeding the results of the neurons to next layer; the components of the final hidden layer then go on to form the output. Unlike single perceptrons, multilayer perceptrons can handle non-linear data and represent arbitrary decision boundaries.

To calculate the weights for each layer, we first compute the errors at the output layer with a cost function. These errors are then distributed backwards from each layer to the previous layer. Then we use gradient descent on the error functions to adjust weight at the layer, so we need all functions differentiable; the weights that minimize the error function form the solution to the weights for the perceptron.

Once we have our vectorized data ready to go, we can easily apply `MLPClassifier()` to our training set then evaluate it on the testing set. Our libraries already have built-in functions to report various metrics, which are expanded more in the next section. For now, we only need to look at accuracy.

```
tfidf_mlp_clf = MLPClassifier(random_state=1, max_iter=100).fit(X_train_tfidf_vector,␣
↪y_train)
y_pred = tfidf_mlp_clf.predict(X_test_tfidf_vector)
tfidf_mlp_clf.score(X_test_tfidf_vector, y_test)
print(metrics.classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           1       0.78      0.74      0.76       481
           2       0.50      0.68      0.58       316
           3       0.77      0.84      0.81       674
           4       0.84      0.83      0.83       863
           5       0.81      0.85      0.83       920
           6       0.80      0.82      0.81       465
           7       0.78      0.81      0.80       730
           8       0.54      0.38      0.45       353
           9       0.54      0.56      0.55       328
          10       0.51      0.47      0.49       256
          11       0.65      0.67      0.66       462
          12       0.67      0.51      0.58       217
          13       0.77      0.74      0.76       443
          14       0.68      0.66      0.67       263
          15       0.76      0.68      0.72       313
          16       0.94      0.93      0.93      1057
```

(continues on next page)

(continued from previous page)

```
    accuracy                           0.76      8141
   macro avg       0.71      0.70      0.70      8141
weighted avg       0.76      0.76      0.76      8141
```

Exercise 1

Repeat the above classification on unigrams and report the accuracy.

Naive Bayes Algorithms

Naive Bayes Algorithms are so called as they rely on Bayes' Theorem and assume high levels of independence (naive). Bayes' Theorem states that

$P(y|X) = \frac{P(X|y)*P(y)}{P(X)}$.

In terms of Bayesian Probability, $P(y|X)$ is called the *posterior*, $P(X|y)$ is called the *data likelihood*, $P(y)$ is the *prior*, and $P(X)$ is the *normalization*. In the context of NLP classification, we define X to be the document features and y to be the document's class label. We also assume conditional independence, which means that the class label of one document does not depend on the class label of another document.

We define $\arg\max_a f$ to be the value of a such that the function f is maximized. Then, Naive Bayes Algorithms classify a document into a class $\hat{y}$ by calculating

$\hat{y} = \arg\max_y P(y|X) = \arg\max_y P(X|y) * P(y)$.

Note that when using Naive Bayes, we have the assumption that features are independent given class. This allows us to compute

$P(x_1, x_2, ..., x_n|y) = \Pi_{i=1}^n P(x_i|y)$

We can estimate either side of this equation with training data, but estimating the right hand side $\Pi_{i=1}^n P(x_i|y)$ leads to a much greater reduction of computation.

Similar to the perceptron, we fit the Naive Bayes algorithm to our training and testing set and report metrics:

```
tfidf_multinomialNB_clf = MultinomialNB().fit(X_train_tfidf_vector, y_train)
y_pred = tfidf_multinomialNB_clf.predict(X_test_tfidf_vector)
print(metrics.classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           1       0.74      0.74      0.74       481
           2       0.85      0.44      0.58       316
           3       0.82      0.84      0.83       674
           4       0.68      0.90      0.78       863
           5       0.63      0.89      0.74       920
           6       0.86      0.81      0.83       465
           7       0.61      0.89      0.72       730
           8       0.81      0.11      0.19       353
           9       0.83      0.23      0.37       328
          10       0.87      0.23      0.36       256
          11       0.78      0.61      0.68       462
          12       0.93      0.29      0.44       217
          13       0.78      0.74      0.76       443
```

(continues on next page)

(continued from previous page)

```
           14       0.93      0.35      0.51       263
           15       0.91      0.50      0.64       313
           16       0.68      0.98      0.80      1057

     accuracy                           0.71      8141
    macro avg       0.79      0.60      0.62      8141
 weighted avg       0.75      0.71      0.68      8141
```

Exercise 2

Repeat the above classifications on unigrams and report the accuracy.

Ridge Classification

Ridge Classification was designed with the original intent of a restriction on basic multiple linear regression; in essence, it adds an additional penalty for extremely large errors, so ridge regression minimizes more than just the least square residuals. However, ridge algorithms can also be used for classification, and in the context of NLP tasks, ridge classifications have been found to perform very well.

In the binary case, ridge converts the two classes to either 1 or -1 and then treats the problem as a regression problem. This can then be generalized to multiple classes with a One vs. Rest approach, where we make an algorithm to determine if an item is in one specific class or any of the rest of the classes; this is then repeated until any item can be classified into a single class.

More on the ways ridge regression is performed and calculated for classification can be found on the scikit-learn explanation: https://scikit-learn.org/stable/modules/linear_model.html#ridge-regression.

Again, similar to previous classification methods, we run a ridge classification as follows:

```
from sklearn.linear_model import RidgeClassifier

tfidf_ridge_clf = RidgeClassifier(tol=1e-2, solver="sparse_cg")
tfidf_ridge_clf = tfidf_ridge_clf.fit(X_train_tfidf_vector, y_train)
y_pred = tfidf_ridge_clf.predict(X_test_tfidf_vector)
print(metrics.classification_report(y_test,y_pred, digits = 4))
```

```
              precision    recall  f1-score   support

           1     0.7160    0.7651    0.7397       481
           2     0.6542    0.6646    0.6593       316
           3     0.8196    0.8427    0.8310       674
           4     0.7558    0.8714    0.8095       863
           5     0.8287    0.8728    0.8502       920
           6     0.8239    0.8151    0.8195       465
           7     0.7995    0.8247    0.8119       730
           8     0.5738    0.3853    0.4610       353
           9     0.5801    0.5518    0.5656       328
          10     0.5990    0.4492    0.5134       256
          11     0.7111    0.6926    0.7018       462
          12     0.7073    0.5346    0.6089       217
          13     0.7817    0.7923    0.7870       443
          14     0.7445    0.6426    0.6898       263
          15     0.7857    0.6677    0.7219       313
          16     0.8920    0.9612    0.9253      1057
```

(continues on next page)

(continued from previous page)

```
              precision    recall  f1-score   support

    accuracy                         0.7732      8141
   macro avg     0.7358    0.7084    0.7185      8141
weighted avg     0.7670    0.7732    0.7676      8141
```

Exercise 3

Repeat the above classification on unigrams and report the accuracy.

14.5.3 More Exercises

Exercise 4

Write a function that takes a document corpus, processed like in the exercises in Section 2, splits the set into a training and testing set, and runs a classifier on the training set then reports metrics on the testing set.

- Your function should take a parameter `classifier_algorithm` that specifies the algorithm to use as well as the parameters necessary for that algorithm.

Exercise 5

Run your function on the SDG corpus using Multinomial Naive Bayes, Multilayer Perceptron, and Ridge Regression.

- Use `min_df` to modify the computation time for Multilayer Perceptron; this will reduce the amount of time it takes to run.

14.6 Evaluating Classification

In many classification problems, the metrics we use include precision, recall. f1, accuracy, and average accuracy (both macro and micro). Additional common metrics for classification include ROC (Receiver Operating Characteristics) and AUC (Area Under the Curve), often combined as ROC-AUC, and for multi-class classification, we can test the model's ability to classify items on a One vs. Rest (OvR) or One vs. One (OvO) basis.

In Section 3, we defined precision, recall, and f1. Accuracy is self-explanatory as the proportion of correctly predicted classes as a whole. The micro average accuracy calculates the average accuracy per class, while macro average accuracy similarly calculates this but takes class imbalance into account.

For this section, we're going to focus on the results from our ridge classification in the previous section. Notice that we get the specific results for each goal on its own, in addition to a confusion matrix for the specific results on the testing set.

```
docs = text_df.embedding.tolist()
scaler = preprocessing.MinMaxScaler().fit(docs)
X = scaler.transform(docs)
y = text_df.sdg
X_train, X_test, y_train, y_test = \
    train_test_split(X, y, test_size=0.33, random_state=7)
ridge_clf = RidgeClassifier(tol=1e-2, solver="sparse_cg")
```

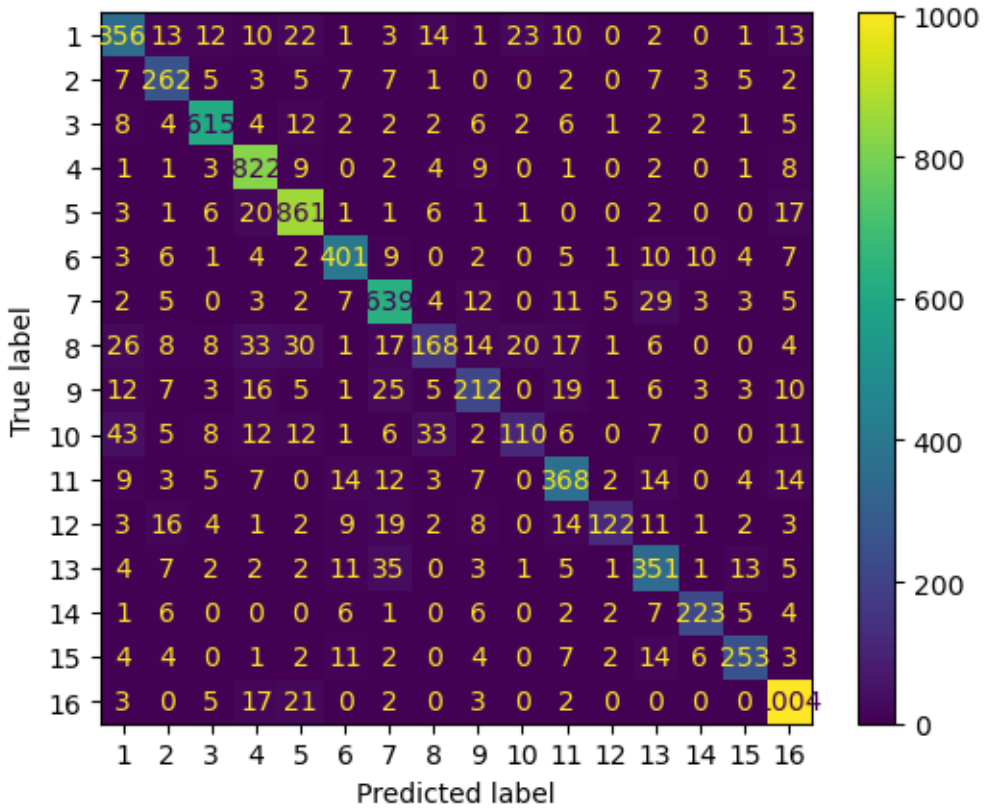
(continues on next page)

(continued from previous page)

```
ridge_clf = ridge_clf.fit(X_train, y_train)
y_pred = ridge_clf.predict(X_test)
print(metrics.classification_report(y_test,y_pred, digits = 4))
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
```

```
              precision    recall  f1-score   support

           1     0.7340    0.7401    0.7371       481
           2     0.7529    0.8291    0.7892       316
           3     0.9084    0.9125    0.9104       674
           4     0.8607    0.9525    0.9043       863
           5     0.8723    0.9359    0.9030       920
           6     0.8478    0.8624    0.8550       465
           7     0.8171    0.8753    0.8452       730
           8     0.6942    0.4759    0.5647       353
           9     0.7310    0.6463    0.6861       328
          10     0.7006    0.4297    0.5327       256
          11     0.7747    0.7965    0.7855       462
          12     0.8841    0.5622    0.6873       217
          13     0.7468    0.7923    0.7689       443
          14     0.8849    0.8479    0.8660       263
          15     0.8576    0.8083    0.8322       313
          16     0.9004    0.9499    0.9245      1057

    accuracy                         0.8312      8141
   macro avg     0.8105    0.7761    0.7870      8141
weighted avg     0.8276    0.8312    0.8257      8141
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x14ed10310>
```



14.6.1 The ROC Curve

The goal of any classifier is to maximize true positive rate (TPR) while minimizing false positive rate (FPR). The ROC curve is then determined as follows: for any given threshold, we line up the points, with TPR on the Y-axis and FPR on the X-axis. Points at the top left corner of the plot imply an FPR of 0 and a TPR of 1 (i.e., perfect classification).

A model can be evaluating qualitatively by judging the "steepness" of its ROC curve; this evaluation can be made quantitative by finding the Area Under the Curve (AUC). A larger AUC implies a steeper curve and is *usually* better.

For binary classification, each point on the ROC curve represents a different threshold and can be a choice for the final classifier to be used; the choice made typically depends on the business requirements and constraints.

14.6.2 Multiclass Classifiers

In any multiclass classification problem, there are multiple class labels $(c_1, c_2, ..., c_k)$ in the data, and each training sample is labeled as belonging to one class only. The strategies used with binary classifiers can be taken in multiclass classifiers by reducing the problem to binary classification. There are two strategies typically taken to do this, those being One vs. Rest (OvR) and One vs. One (OvO).

One vs Rest

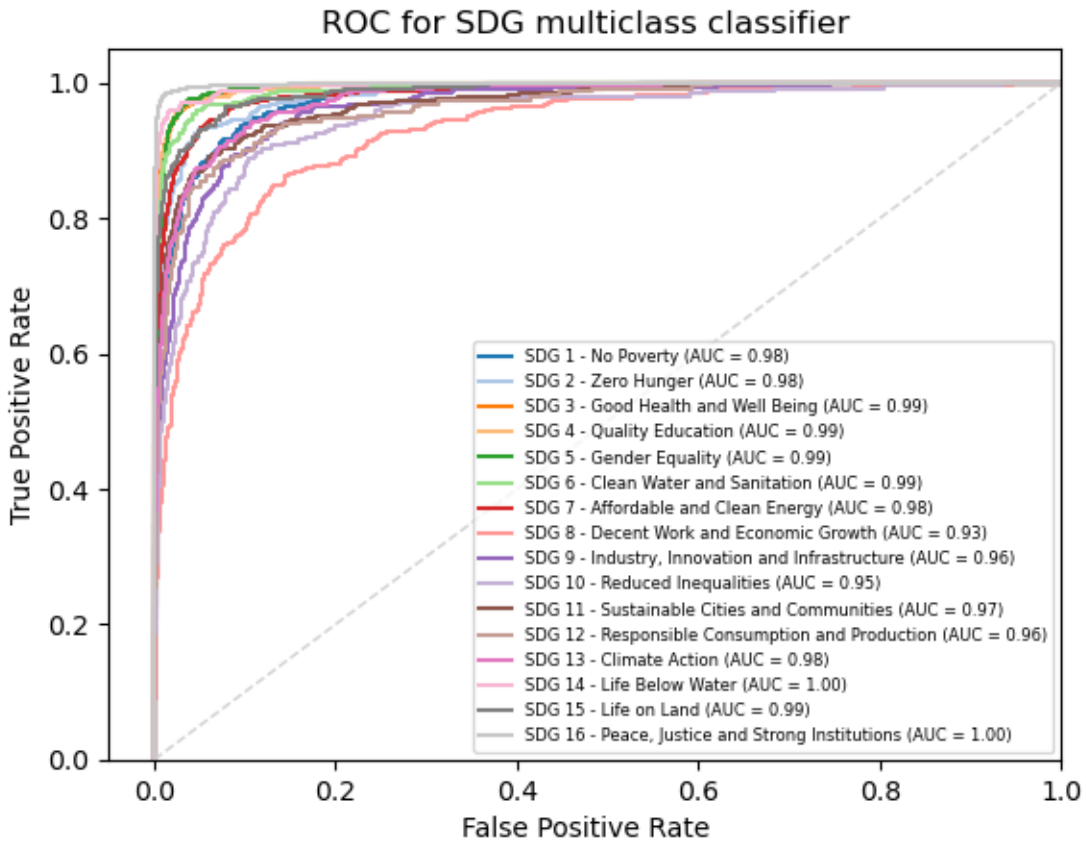
The OvR method of evaluation judges the model ability to determine if a given item belongs to a given class or if it is in one of the other classes. For each class label c in $(c_1, c_2, ..., c_k)$, we say that the *positive samples* are those that are labeled as class c, and the *negative samples* are all the rest of the samples. We then fit and predict for class c.

For scoring each sample, we simply take the highest probable score (i.e., highest score) of the k classifiers as the class for the sample.

When looking at the ROC curve for the OvR strategy on our classification, we get the following graphic:

```
docs = text_df.embedding.tolist()
scaler = preprocessing.MinMaxScaler().fit(docs)
X = scaler.transform(docs)
y = text_df.sdg

label_binarizer = LabelBinarizer().fit(y)
y_onehot = label_binarizer.transform(y)
n_classes = len(label_binarizer.classes_)
class_names = [sdg_names[sdg_names["sdg"] == label_binarizer.classes_[i]].sdg_name.
 ↪item() \
               for i in range(n_classes)]

X_train, X_test, y_train, y_test = train_test_split(X, y_onehot, test_size=.33,␣
 ↪random_state=0)
ovr_mlp_clf = OneVsRestClassifier(MLPClassifier(random_state=0, max_iter=300)).fit(X_
 ↪train,y_train)
y_score = ovr_mlp_clf.predict_proba(X_test)
```

```
fpr = dict()
tpr = dict()
roc_auc = dict()
for class_id in range(n_classes):
    fpr[class_id], tpr[class_id], _ = metrics.roc_curve(y_test[:, class_id], y_
 ↪score[:, class_id]) # roc_curve works on binary
    roc_auc[class_id] = metrics.auc(fpr[class_id], tpr[class_id])

for class_id, color in zip(range(n_classes), colors):
    plt.plot(fpr[class_id], tpr[class_id], color=color, lw=1.5,alpha = 1,
             label='SDG {0} - {1} (AUC = {2:0.2f})'
             ''.format(class_id+1, class_names[class_id], roc_auc[class_id]))
plt.plot([0, 1], [0, 1], '--', lw=1, color="lightgrey")
plt.xlim([-0.05, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC for SDG multiclass classifier')
plt.legend(loc="lower right", fontsize=6)
plt.show()
```


ROC for SDG multiclass classifier
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
SDG 1 - No Poverty (AUC = 0.98)
SDG 2 - Zero Hunger (AUC = 0.98)
SDG 3 - Good Health and Well Being (AUC = 0.99)
SDG 4 - Quality Education (AUC = 0.99)
SDG 5 - Gender Equality (AUC = 0.99)
SDG 6 - Clean Water and Sanitation (AUC = 0.99)
SDG 7 - Affordable and Clean Energy (AUC = 0.98)
SDG 8 - Decent Work and Economic Growth (AUC = 0.93)
SDG 9 - Industry, Innovation and Infrastructure (AUC = 0.96)
SDG 10 - Reduced Inequalities (AUC = 0.95)
SDG 11 - Sustainable Cities and Communities (AUC = 0.97)
SDG 12 - Responsible Consumption and Production (AUC = 0.96)
SDG 13 - Climate Action (AUC = 0.98)
SDG 14 - Life Below Water (AUC = 1.00)
SDG 15 - Life on Land (AUC = 0.99)
SDG 16 - Peace, Justice and Strong Institutions (AUC = 1.00)

We can see that each of the goals, aside from goal 17, gets its own curve, and our program is able to report the AUC score for each one.

Exercise 1

Create a similar ROC curve plot for your Naive Bayes classification and Multilayer Perceptron classification from the previous section.

One vs One

The OvO approach compares each class to another single class for all possible pairs of classes. For each class label c in $(c_1, c_2, ..., c_k)$, and for each pair (c, c_j), with $c_j \neq c$, we say that the *positive samples* are those that are labeled as class c, and the *negative samples* are those that are labeled as class c_j.

For each sample to be scored, it gets $k - 1$ classification results, and we then vote by majority to determine the class for the sample.

Similar to the OvR strategy, we can produce an ROC curve for the OvO strategy as follows:

```
docs = text_df.embedding.tolist()
scaler = preprocessing.MinMaxScaler().fit(docs)
X = scaler.transform(docs)
y = text_df.sdg

X_train, X_test, y_train, y_test = \
```

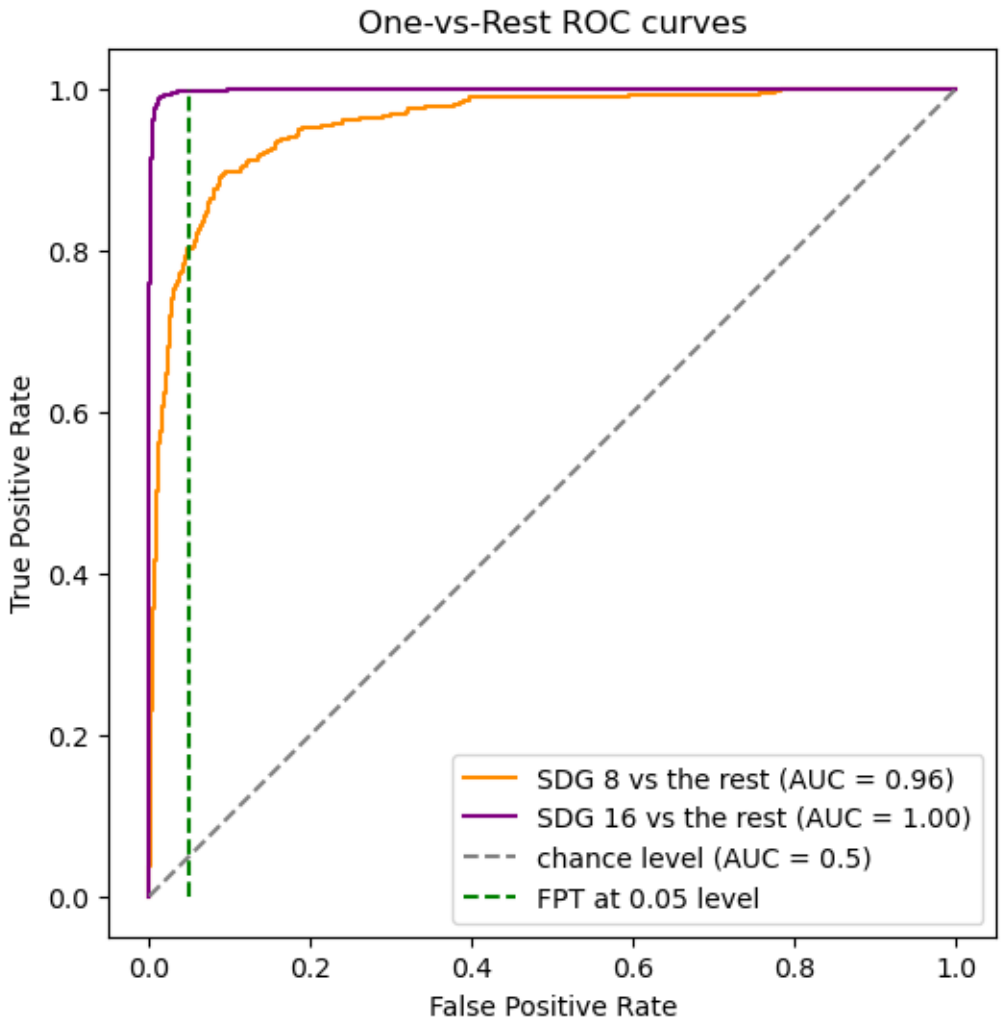
(continues on next page)

(continued from previous page)

```
    train_test_split(X, y, test_size=0.33, random_state=7)

logit_clf = LogisticRegression(tol=1e-2, solver="liblinear")
logit_clf = logit_clf.fit(X_train, y_train)

# metrics.roc_auc_score work on binary and multiclass
y_score = logit_clf.predict_proba(X_test)
print(metrics.roc_auc_score(y_test,y_score, multi_class='ovr'))
```

```
0.9830440625522275
```

```
label_binarizer = LabelBinarizer().fit(y)
y_onehot_test = label_binarizer.transform(y_test)

fig, ax = plt.subplots(figsize=(6,6))

class_of_interest = 8 # SDG 8
class_id = np.flatnonzero(label_binarizer.classes_ == class_of_interest)[0]
RocCurveDisplay.from_predictions(
    y_onehot_test[:, class_id],
    y_score[:, class_id],
    name=f"SDG {class_of_interest} vs the rest",
    color="darkorange",
    ax = ax,
)

class_of_interest = 16
class_id = np.flatnonzero(label_binarizer.classes_ == class_of_interest)[0]
RocCurveDisplay.from_predictions(
    y_onehot_test[:, class_id],
    y_score[:, class_id],
    name=f"SDG {class_of_interest} vs the rest",
    color="purple",
    ax = ax,
)

plt.plot([0, 1], [0, 1], "--", label="chance level (AUC = 0.5)", color = "grey")
plt.plot([0.05, 0.05], [0, 1], "--", label="FPT at 0.05 level", color = "green")

plt.axis("square")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("One-vs-Rest ROC curves")
plt.legend()
plt.show()
```



Here, we identified two classes that which we wanted to specifically examine, and these two are the only curves that are present on the plot. Note that the AUC for these goals we picked are different from those in the OvR strategy.

Exercise 2

Create another ROC curve plot for three additional goals of your choosing.

Exercise 3

Using SDGs 8 and 16, recreate the ROC curve plot using your Naive Bayes and Multilayer Perceptron classifications from the previous sections.

14.6.3 More Exercises

Exercise 4

Modify your function from Exercise 4 in the previous section to now output an OvR ROC plot, which can be toggled with a parameter `roc_plot`.

Additionally, add a parameter `quick_metrics` that will not output the entire table of metrics and only the overall accuracy, precision, recall, and F1-Score.

Exercise 5

What does a point on the ROC curve at (1,1) indicate about the corresponding model? Similarly, what does a point at (0,0) on the curve indicate about the model?

14.7 Applications

In this chapter, we have covered some of the basics of text problems and language processing, we also covered how to build text classifier by different approaches such as n-gram based vectorizer, and text embedding.

In this last section of the chapter, we will first show an example of using the text classifier to classify newly acquired text content. We will then show a use of the text features for the problem of text data retrieval.

14.7.1 Classifying a given text

One common method for acquiring new text data to feed into a text classifier is called web scraping. We will use a convenient tool called `BeautifulSoup` to scrape the content of a web page, then run the scraped text content through the UNSDG classifier that we have built over the previous sections. `BeautifulSoup` provides a functionality to parse a web request response into HTML elements with their tags (i.e., it breaks down any web page into its smaller sections, like the paragraphs, images, etc). Searching through the paragraph tags will grab the majority of the content of a web page.

To get started, we first import the `requests` library and `BeautifulSoup` itself. We will then go to the official UN website and read in one of their news stories.

```
import requests
from bs4 import BeautifulSoup
url = "https://news.un.org/en/story/2023/07/1138767"
response = requests.get(url)
soup = BeautifulSoup(response.content, "html.parser")
```

We then use the NLTK sentence tokenizer to break the web page content into sentences.

```
from nltk import sent_tokenize
page_sentences = []
for tag in soup.find_all('p'):
        if (len(tag.text)>=30):
            sentences = sent_tokenize(tag.text)
            page_sentences = page_sentences + sentences
page_sentence_df = pd.DataFrame({"text": page_sentences})
page_sentence_df.text
```

```
0     New data from the World Health Organization (W...
1     Global immunisation services reached four mill...
2     Data published by the UN agencies on Monday re...
3     DTP vaccinations are commonly used as the glob...
4     Despite the improvement, this figure is still ...
5     “These data are encouraging, and a tribute to ...
6     “But global and regional averages don’t tell t...
7     When countries and regions lag, children pay t...
8     The early stages of recovery in immunisation r...
9     Progress in well-resourced countries with larg...
10    Of the 73 countries that recorded substantial ...
11    Vaccination against measles, one of the most i...
12    Last year, 21.9 million children - 2.7 million...
13    This has placed children in under-vaccinated c...
14    Data indicates countries with sustained immuni...
15    South Asia, which reported gradual increases i...
16    The African region, which is lagging behind in...
17    As population size increases, countries must s...
18    With support from Gavi, the Vaccine Alliance, ...
19    The increase in DTP3 coverage in Gavi-implemen...
20    "It is incredibly reassuring, after the massiv...
21    “However, it is also clear from this important...
22    For the first time since the COVID-19 pandemic...
23    HPV vaccination programmes that began pre-pand...
24    Many stakeholders are working to improve routi...
25    In 2023, WHO and UNICEF, along with Gavi, The ...
26    The movement aims to secure financing for immu...
27    “Beneath the positive trend lies a grave warni...
28    “Until more countries mend the gaps in routine...
29       Viruses like measles do not recognise borders.
30    Efforts must urgently be strengthened to catch...
31    Thank you in advance for agreeing to participa...
32    The survey will take no more than 4 minutes to...
Name: text, dtype: object
```

Using these sentences, we could train a classification model like we did in Section 5, but we could also increase the training portion of labeled data, say 90% of the labeled data for training and only 10% just to get a confirmative check on the model.

Exercise 1

Why is it the case that we can use more data for training in this case?

First we read in the training data. Notice that we put the scripts used in previous sections into a function. As with previous sections, make sure to adjust the loading of data to fit your directories.

```
# change this to your own data directory
data_dir = "data/"

# read and preprocess data
text_file_name = "osdg-community-data-v2023-01-01.csv"
text_df = pd.read_csv(data_dir + text_file_name,sep = "\t",  quotechar='"')
col_names = text_df.columns.values[0].split('\t')
text_df[col_names] = text_df[text_df.columns.values[0]].apply(lambda x: pd.
↪Series(str(x).split("\t")))
```

(continues on next page)

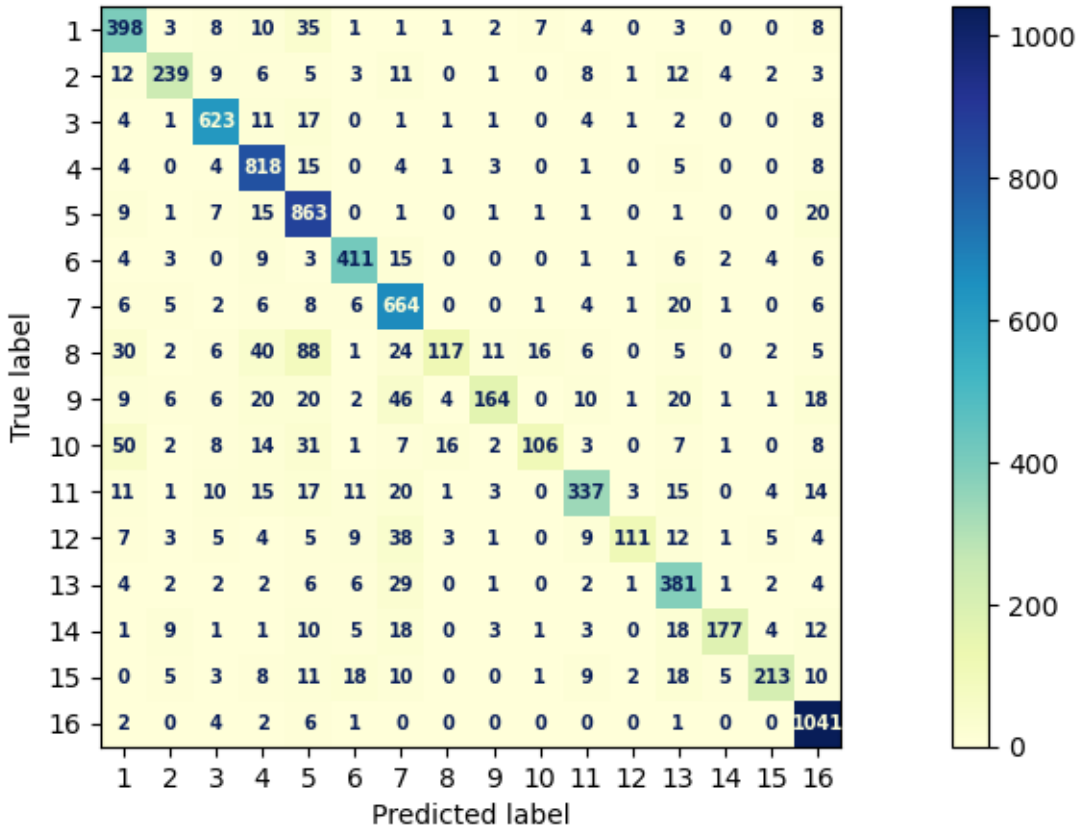
(continued from previous page)

```
text_df = text_df.astype({'sdg':int, 'labels_negative': int, 'labels_positive':int,
 ↪'agreement': float}, copy=True)
text_df.drop(text_df.columns.values[0], axis=1, inplace=True)
text_df = text_df.query("agreement > 0.5 and (labels_positive - labels_negative) > 2")
text_df.reset_index(inplace=True, drop=True)
```

We then train a classifier using `CountVectorizer` to generate the features vector and train a multinomial Naive Bayes classifier.

```
docs = text_df.text
categories = text_df.sdg
X_train, X_test, y_train, y_test = \
    train_test_split(docs, categories, test_size=0.33, random_state=7)

X_train_count_vectorizer = CountVectorizer(ngram_range=(2,2), stop_words = "english" )
X_train_count_vectorizer.fit(X_train)
X_train_count_vectorizer = X_train_count_vectorizer.transform(X_train)
X_test_count_vector = X_train_count_vectorizer.transform(X_test)

count_multinomialNB_clf = MultinomialNB().fit(X_train_count_vector, y_train)
y_pred = count_multinomialNB_clf.predict(X_test_count_vector)

fig, ax = plt.subplots(figsize=(15, 5))
font = {'family': 'sans-serif', 'weight': 'heavy','size': 7,} # set font for␣
 ↪displaying confusion matrix
cm = mpl.colormaps["YlGnBu"] # set the color map for displaying confusion matrix
ConfusionMatrixDisplay.from_predictions(y_test, y_pred, text_kw=font, ax=ax, cmap=cm)
print(metrics.classification_report(y_test,y_pred, digits = 4))
```

```
              precision    recall  f1-score   support

           1     0.7223    0.8274    0.7713       481
           2     0.8475    0.7563    0.7993       316
           3     0.8926    0.9243    0.9082       674
           4     0.8338    0.9479    0.8872       863
           5     0.7570    0.9380    0.8379       920
           6     0.8653    0.8839    0.8745       465
           7     0.7469    0.9096    0.8203       730
           8     0.8125    0.3314    0.4708       353
           9     0.8497    0.5000    0.6296       328
          10     0.7970    0.4141    0.5450       256
          11     0.8383    0.7294    0.7801       462
          12     0.9098    0.5115    0.6549       217
          13     0.7243    0.8600    0.7864       443
          14     0.9171    0.6730    0.7763       263
          15     0.8987    0.6805    0.7745       313
          16     0.8860    0.9849    0.9328      1057

    accuracy                         0.8184      8141
   macro avg     0.8312    0.7420    0.7656      8141
weighted avg     0.8248    0.8184    0.8074      8141
```



First, we run the classifier as if each sentence is a document and will be classified into its respective UNSDG. We can see the results indicate that more sentences are about SDG 3, which is "Good Health and Well Being", SDG 1 "No Poverty", and SDG 5 "Gender Equality".

```
page_count_vector = X_train_count_vectorizer.transform(page_sentence_df.text)
page_pred = count_multinomialNB_clf.predict(page_count_vector)
pd.DataFrame(page_pred).value_counts()
```

```
3     10
1      8
5      7
16     5
4      2
11     1
dtype: int64
```

Exericse 6.2

Repeat the above classification of the new data with Ridge and Multilayer Perceptron classifications. What differences exist?

We can put all sentences of the web page content into one text document and classify the entire page content into SDG. We can see the result shows the web page is about SDG 3, "Good Health and Well Being".

```
page_count_vector = X_train_count_vectorizer.transform(pd.Series(page_sentence_df.
```

(continues on next page)

(continued from previous page)

```
↪text.str.cat()))
page_pred = count_multinomialNB_clf.predict(page_count_vector)
page_pred
```

```
array([3])
```

Exercise 3

Read the news article at https://www.rockefellerfoundation.org/commitment/economic-equity/. What goal would you classify it as? What goal does the Naive Bayes model classify the document and each sentence as?

14.7.2 Similarity Retrieval

In essence, similarity retrieval is the process of item retrieval based on similarity. It is most common and familiar in the context of product recommendation for online shopping, but it can be used for documents as well where each document is simply treated as an item.

The problem itself can be boiled down to **estimating a utility function that predicts how much a user will "like" an item**; this terminology derives from online retail recommendations, but a very similar process exists for document similarity. In the context of documents, this utility would most likely be measured in terms of engagement with the document (for example, how long the user spent reading it, or if they read it at all), and we would most likely build recommendations purely based off of content similarity.

The most common way to perform this task is to start by making a **document-token interaction matrix**. In these matrices, each document in our dataset corresponds to one row, and each token corresponds to a column. Each entry (x_i, y_j) of the matrix then corresponds to the number of instances of token j in document i.

To define similarity, then, we can treat each row or column as a vector and look at how similar any two vectors are. Note that we would use row vectors to determine similarity between documents and column vectors to determine similarity between token occurrences. Similarity is then mathematically computed as **cosine similarity**, where for any two vectors $\vec{A}$ and $\vec{B}$, the similarity is the cosine of the angle between them, or

$$\cos(\theta) = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\| \times \|\vec{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Exericse 6.4

Would high similarity indicate a lower or higher cosine similarity score? Why?

To implement this in our code, we first return to the `tensorflow` package.

```
import tensorflow as tf
import tensorflow_hub as hub

file_name = "[your-directory]/osdg-community-data-v2023-01-01.csv"
text_df = get_text_df(file_name)

embed = hub.load("[your-directory]/universal-sentence-encoder_4")

text_df["embedding"] = list(embed(text_df.text))
```

```
import tensorflow as tf
import tensorflow_hub as hub

# change this to your own embedding directory
embedding_dir = "embeddings/"

# load the embedding
embed = hub.load(embedding_dir + "universal-sentence-encoder_4")

text_df["embedding"] = list(embed(text_df.text))
```

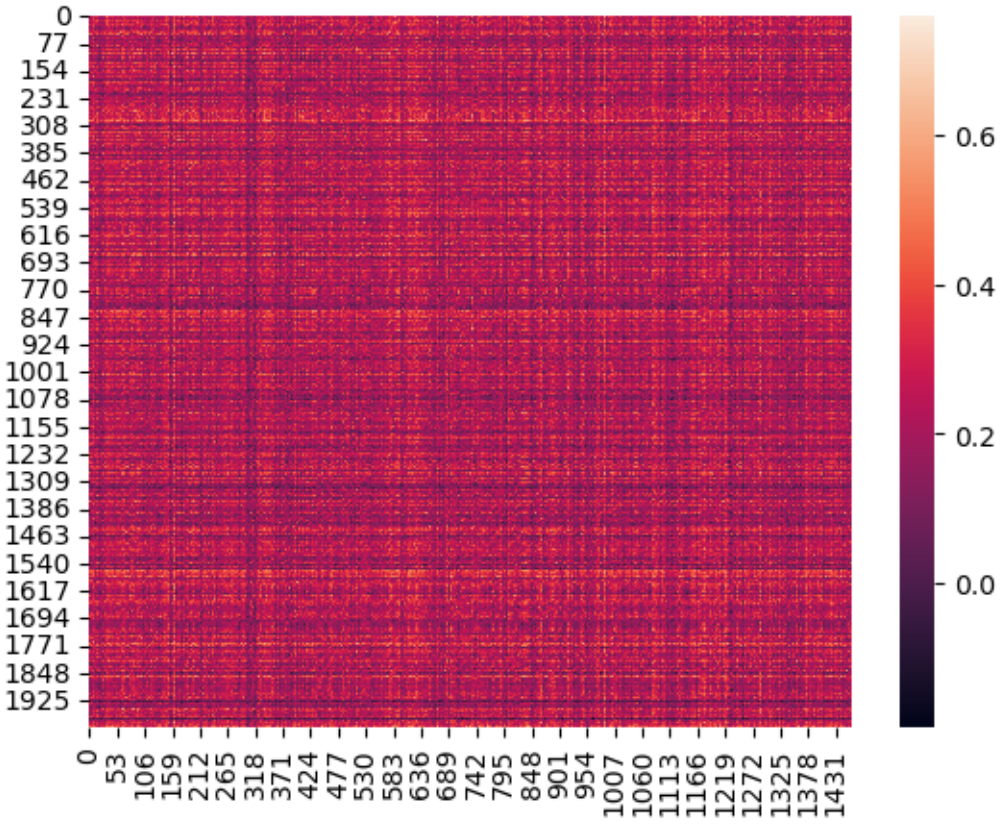
We will also make a dataframe with all of the sentences:

```
def get_sentence_df(text_df):
    text_df_sentence = []
    text_df_sdg = []
    text_df_text_id = []
    embedding = []
    for (text, sdg, text_id) in iter(zip(text_df.text, text_df.sdg, text_df.text_id)):
        temp_sentence = sent_tokenize(text)
        text_df_sentence = text_df_sentence + temp_sentence
        text_df_sdg = text_df_sdg + [sdg]*len(temp_sentence)
        text_df_text_id = text_df_text_id + [text_id]*len(temp_sentence)
        embedding = embedding + list(embed(temp_sentence))
    sentence_df = pd.DataFrame({"text": text_df_sentence,
                                "sdg": text_df_sdg,
                                "text_id": text_df_text_id,
                                "embedding": embedding})
    return sentence_df

sentence_df = get_sentence_df(text_df)
```

The process of generating similarity will result in a similarity heatmap as shown below. This heatmap can be thought of as a square matrix where each row and column represents a document, and the color of each entry represents the similarity between the documents.

```
sns.heatmap(np.array(np.inner(text_df.query("sdg == 3").embedding.tolist(), \
                              text_df.query("sdg == 1").embedding.tolist())))
```

```
<Axes: >
```

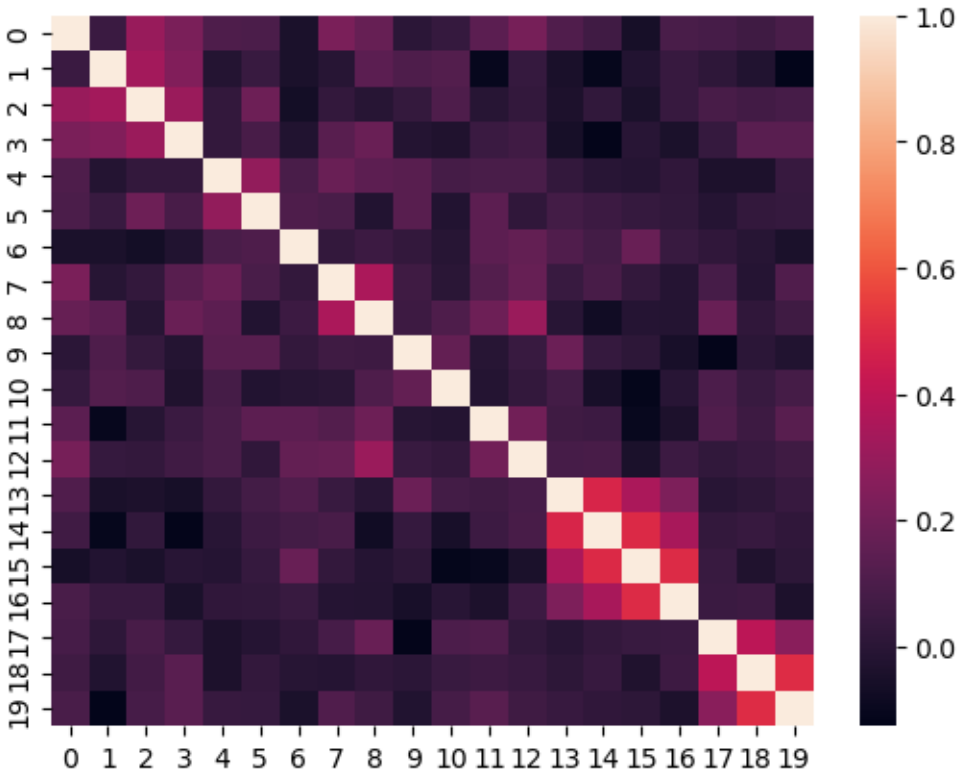


Since we have a lot of documents, it might not make much sense to make a huge heatmap as it is not entirely visible.

Additionally, we can also look at heatmaps for sentence similarity. Let's look at the similarities between the first 20 sentences in the dataset.

```
sns.heatmap(np.array(np.inner(sentence_df.embedding[:20].tolist(), \
                              sentence_df.embedding[:20].tolist())))
```

```
<Axes: >
```



Exercise 5

Modify the code to generate a heatmap of only the first 40 documents.

Exercise 6

(Bonus) Generate the heatmap for all the sentences. This may take a while and should be done with a beefy computer.

If we want to actually return the documents that are most similar to one that we choose (or any sentence of our choosing), we write a function that returns the n most similar documents or sentences:

```
def get_most_similar(text_df, sentence, n = 5):
    sentence_sim = np.inner(list(text_df.embedding), embed([sentence]))
    val = sorted(sentence_sim, reverse=True)[n]
    return text_df[sentence_sim > val].text
```

We can then test our function by running it on an example sentence, starting by looking at the most similar sentences from our dataset:

```
s = "countries are working hard to save the ocean aminals."
get_most_similar(sentence_df, sentence = s, n=5)
```

```
1566      For indeed, there is no agency for ocean issues.
16750    As an island nation surrounded by the sea, we ...
78771    I think no country in the world can protect th...
```

(continues on next page)

(continued from previous page)

```
86262    The rich countries also have a responsibility ...
87077    Governments investing in marine biotechnology ...
Name: text, dtype: object
```

Exercise 7

What *documents* are the most similar to our example sentence?

Exercise 8

Generate a similarity heatmap for the *sentences* classified as SDG 1 and SDG 3.

14.7.3 More Exercises

Exercise 9

Write a function that takes a URL website as a string and returns the most similar documents in our dataset.

Exercise 10

Run the function from Exercise 9 on the following websites:

- http://gianttortoise.org/en/beyond-tracking
- https://www.dhs.gov/blue-campaign/what-human-trafficking
- https://www.dol.gov/agencies/odep/program-areas/individuals/older-workers
- https://michigantoday.umich.edu/2022/08/26/positively-breaking-the-age-code/

Do the most similar documents seem reasonable?

14.8 Solutions to Exercises: Sections 1 to 4

14.8.1 Preprocessing

Exercise 1

Answers may vary.

Exercise 2

Answers may vary.

Exercise 3

The following code removes any rows that contain only N/A values. In this case, there are no such rows to remove.

```
nrows_old = text_df.shape[0]
text_df.dropna(axis=0, how='all', inplace=True)
print("Number of rows removed:", nrows_old - text_df.shape[0])
```

```
Number of rows removed: 0
```

The next line of code checks for the existence of any remaining N/A values. It turns out that there are none.

```
text_df.isna().any()
```

```
doi                False
text_id            False
text               False
sdg                False
labels_negative    False
labels_positive    False
agreement          False
dtype: bool
```

Whether or not entries with N/A values should be removed depends on the dataset and the nature of the problem. Sometimes, entries with N/A values should be dropped, while at other times, they should be kept unchanged, or replaced with interpolated or placeholder values. Consult the `pandas` documentation for more information about how to deal with missing values in dataframes.

Exercise 4

After filtering the dataset, we inspect it using the `info()` function.

```
# filter the dataset
text_df = text_df.query("agreement > 0.5 and (labels_positive - labels_negative) > 2")
text_df.reset_index(inplace=True, drop=True)

# inspect it
text_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24669 entries, 0 to 24668
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   doi              24669 non-null  object
 1   text_id          24669 non-null  object
 2   text             24669 non-null  object
 3   sdg              24669 non-null  int64
 4   labels_negative  24669 non-null  int64
 5   labels_positive  24669 non-null  int64
 6   agreement        24669 non-null  float64
dtypes: float64(1), int64(3), object(3)
memory usage: 1.3+ MB
```

We have 40062 entries with 7 features (see *section 1* for details). The data types range from `object` (likely denoting strings) to `int64` (integers) to `float64` (floating-point numbers). This is a reasonable amount of data to work with.

Exercise 5

The Porter and Snowball stemmers are largely comparable, while the Lancaster stemmer is the most aggressive. As a result, the Lancaster stemmer is likely to have the most trouble on a larger set of tokens.

Exercise 6

Answers may vary. Some possible observations include the fact that stemmers tend to remove affixes (such as `-ing`, `-ed`, and `-s` in English) and the fact that irregular words are particularly likely to give the stemmers trouble.

Exercise 7

Answers may vary.

Exercise 8

Answers may vary. Some possible entity labels include `GPE` ("nationalities or religious or political groups"), `TIME` ("times smaller than a day"), `QUANTITY` ("measurements, as of weight or distance"), and `WORK_OF_ART` ("titles of books, songs, etc.").

Exercise 9

Sample code solution:

```
# load trained pipeline
nlp = spacy.load('en_core_web_sm')

# perform NER on random sample in both original and lower case
sample = text_df['text'].sample(1).values[0]
doc = nlp(sample)
print('ORIGINAL CASE')
spacy.displacy.render(doc, style='ent', jupyter=True)
print('\nLOWERCASE')
doc = nlp(sample.lower())
spacy.displacy.render(doc, style='ent', jupyter=True)
```

```
ORIGINAL CASE
```

```
<IPython.core.display.HTML object>
```

```
LOWERCASE
```

```
<IPython.core.display.HTML object>
```

Answers may vary depending on the samples chosen. This sample demonstrates that the model sometimes confuses organizations with people. Additionally, it shows that the model often fails to recognize organization names (especially abbreviated ones) when they are converted to lowercase.

Exercise 10

Answers may vary.

14.8.2 About Text Data

Exercise 1

```
# get document-term matrix
docs = text_df.text
cv = CountVectorizer()
cv_fit = cv.fit_transform(docs)

# get feature names and total counts
feature_names = cv.get_feature_names_out()
```

(continues on next page)

(continued from previous page)

```
total_counts = cv_fit.sum(axis=0)

# get the index of the most frequent word
most_freq_feature = total_counts.argmax()

# get the most frequent word itself
most_freq_token = feature_names[most_freq_feature]
print(f"Most frequent word: '{most_freq_token}'")
```

```
Most frequent word: 'the'
```

Exercise 2

```
# get document-term matrix with stop words removed
cv2 = CountVectorizer(stop_words='english') # exclude English stop words
cv2_fit = cv2.fit_transform(text_df.text)

original_len = len(cv.vocabulary_) # length of the original vocabulary (with stop␣
↪words)
new_len = len(cv2.vocabulary_) # length of the new vocabulary (without stop words)
stopwords = cv2.get_stop_words()

print('Length of the original vocabulary (with stop words):', original_len)
print('Length of the new vocabulary (without stop words):', new_len)
print('Number of stop words:', len(stopwords))
print('Difference between original and new vocabularies:', original_len - new_len)
```

```
Length of the original vocabulary (with stop words): 45738
Length of the new vocabulary (without stop words): 45440
Number of stop words: 318
Difference between original and new vocabularies: 298
```

The difference between the original and new vocabularies is less than the number of stop words. This is because not all of the stop words actually occur in the original vocabulary. The following code lists the stop words that are missing from the original vocabulary. Note how the difference between the original and new vocabulary lengths (298) added to the number of missing stopwords (20) is equal to the total number of stop words (318).

```
missing_stopwords = stopwords - cv.vocabulary_.keys()
print(f'{len(missing_stopwords)} missing stopwords:', missing_stopwords)
```

```
20 missing stopwords: {'whereafter', 'whence', 'noone', 'thereupon', 'i', 'thence',
↪ 'a', 'latterly', 'yours', 'whereupon', 'couldnt', 'whoever', 'anyhow', 'hasnt',
↪'whither', 'hers', 'amoungst', 'hereupon', 'yourselves', 'beforehand'}
```

Exercise 3

```
# get feature names and total counts
feature_names = cv2.get_feature_names_out()
total_counts = cv2_fit.sum(axis=0)

# get the index of the most frequent word
most_freq_feature = total_counts.argmax()

# get the most frequent word itself
most_freq_token = feature_names[most_freq_feature]
print(f"Most frequent word: '{most_freq_token}'")
```

```
Most frequent word: 'countries'
```

Exercise 4

First, we fit the one-hot encoder to the sample text.

```
sample_text = text_df.text.iloc[12737].lower()
tokens = nltk.word_tokenize(sample_text)

def ohe_reshape(tokens):
    return np.asarray(tokens).reshape(-1,1)

ohe = OneHotEncoder(handle_unknown='ignore') # encode unknown tokens as vectors of␣
 ↪all zeros
ohe.fit(ohe_reshape(tokens));
```

Next, we transform each token only once by using a set to remove duplicates.

```
token_set = list(set(tokens))
encodings = ohe.transform(ohe_reshape(token_set)).toarray() # encode the tokens
```

There are multiple ways to check that the resulting encodings are unique, but one simple way is to use the `pandas` library. The following code transforms the encodings into a `pandas` dataframe and then verifies that there are no duplicates. This confirms that each learned token has a unique encoding.

```
pd.DataFrame(encodings).duplicated().any()
```

```
False
```

Exercise 5

```
print('SDG:', text_df.sdg.iloc[118])
print('Text:', text_df.text.iloc[118])
```

```
SDG: 5
Text: "Female economic activities were critically examined and new light was shed␣
 ↪on existing conceptions of traditional housework. Oxford University Press, 2007).
 ↪ An edited version of Ihe chapter is available al www.rci.rutgers.edu/~cwgl/
 ↪globalcenler/charlotte/UN-Handbook.pdf. Targets were also set for the␣
 ↪improvement of women's access to economic, social and cultural rights, including␣
 ↪improvements in health, reproductive services and sanitation. The women in␣
 ↪development approach is embodied in article 14 of the Convention, which focuses␣
 ↪on rural women and calls on States to ensure that women ""participate in and␣
 ↪benefit from rural development"" and also that they ""participate in the␣
 ↪elaboration and implementation of development planning at all levels"".15␣
 ↪Participation is an important component of the right to development, as␣
 ↪discussed below."
```

The most frequent words are “women” and “development”, which occur 4 times each. This, together with the label of SDG 5 (gender equality), suggests that this document is about equality for women.

Exercise 6

Each token in a given document, except for the first and last, is grouped into two different bigrams (one with the previous token, and another with the next token). In this case, the large number of distinct bigrams in the entire corpus likely leads to a bigram vocabulary that is larger than the corresponding unigram vocabulary. However, many of the unigrams may occur more often than many of the bigrams do, making the total count of bigrams smaller than the total count of unigrams.

Exercise 7

```
count_vectorizer = CountVectorizer(ngram_range=(3,3), stop_words='english')
count_vector = count_vectorizer.fit_transform(docs)
print('Total count of trigrams (without stop words):', count_vector.sum())
print('Number of unique trigrams (without stop words):', len(count_vectorizer.
 ↪vocabulary_))
```

```
Total count of trigrams (without stop words): 1301713
Number of unique trigrams (without stop words): 1214215
```

The total count of trigrams is smaller than the total count of bigrams, but the number of unique trigrams is larger than the total number of unique bigrams. The explanation for this is similar to the reasoning offered in the solution to the previous exercise, but substituting bigrams for unigrams and trigrams for bigrams.

Exercise 8

Answers may vary depending on the sentences chosen.

Exercise 9

```
tp = 398
fp = 153
fn = 83

precision = tp / (tp + fp)
recall = tp / (tp + fn)
f1 = 2 * (precision * recall)/(precision + recall)

print(f'Precision = {precision}, recall = {recall}, f1 = {f1}')
```

```
Precision = 0.7223230490018149, recall = 0.8274428274428275, f1 = 0.
 ↪7713178294573643
```

Exercise 10

Answers may vary depending on the parameters chosen. Here is a sample answer using the parameters `ngram_range = (2,2)` (for bigrams), `stop_words = 'english'`, and `min_df = 3`.

```
docs = text_df.text
count_vectorizer = CountVectorizer(ngram_range=(2,2), stop_words='english', min_df=3)
count_vector = count_vectorizer.fit_transform(docs).toarray()
term_freq = pd.DataFrame({"term": count_vectorizer.get_feature_names_out(), "freq" :␣
 ↪count_vector.sum(axis=0)})

# find the frequencies of the 5 most common bigrams
term_freq.sort_values(by='freq', ascending=False).iloc[0:5]
```

	term	freq
27774	human rights	1981
10071	climate change	1301
20488	et al	1167
40775	oecd countries	898
26527	health care	881

```
docs = text_df.text
count_vectorizer = CountVectorizer(ngram_range=(2,2), stop_words='english', min_df=3)
```

(continues on next page)

(continued from previous page)

```
count_vector = count_vectorizer.fit_transform(docs).toarray()
count_vector_df = pd.DataFrame(count_vector, columns=count_vectorizer.get_feature_
↪names_out())

# find the frequencies of the 5 most common bigrams
term_freq.sort_values(by='freq', ascending=False).iloc[0:5]
```

Exercise 11

```
def analyze_frequency(corpus, stop_words=None):
    # create term-document matrix
    count_vectorizer = CountVectorizer(ngram_range=(1,1), stop_words=stop_words)
    count_vector = count_vectorizer.fit_transform(corpus).toarray()

    # calculate frequencies of 50 most frequent terms
    freq_df = pd.DataFrame(
        {"term": count_vectorizer.get_feature_names_out(), "freq" : count_vector.
↪sum(axis=0)}
    ).sort_values(by='freq', ascending=False)

    # calculate cumulative word counts
    csum = np.cumsum(freq_df.iloc[0:50].freq).values

    # create plot
    fig, ax = plt.subplots()
    plt.plot(csum)
    ax.set_ylabel('cumulative word count')
    ax.set_xlabel('rank')
    ax.set_title('Cumulative Word Count (Most Frequent to 50th Most Frequent)')

    # calculate comparison
    comp = csum[-1] / freq_df.freq.sum()

    return (freq_df.iloc[0:50], ax, f'{comp:.2%}')
```

Exercise 12

First, we obtain our corpus:

```
sdg8 = text_df[text_df.sdg == 8].text
```

Then we run the function from the previous exercise on that corpus and examine the results:

```
(top_50_words, plot, pct) = analyze_frequency(sdg8)

print('Level of cumulation percentage:', pct)
print()
print('Top 50 words:')
top_50_words
```

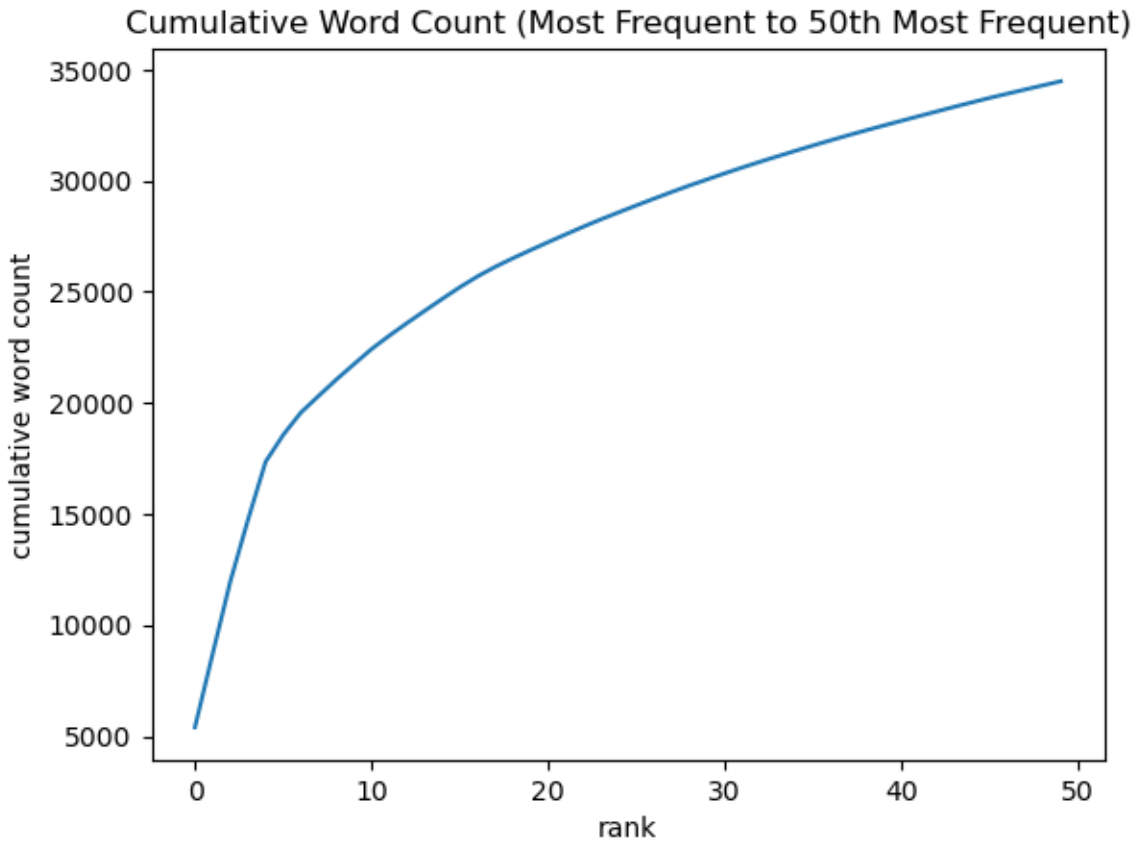
```
Level of cumulation percentage: 38.58%

Top 50 words:
```

```
             term  freq
6933          the  5400
```

(continues on next page)

(continued from previous page)

```
4806           of  3294
636           and  3261
3578           in  2777
7000           to  2618
3018          for  1212
3842           is  1005
6932         that   745
712           are   734
4843           on   694
747            as   683
7546         with   601
2498   employment   566
927            be   550
1137           by   539
3994       labour   530
6958         this   485
7565      workers   430
4888           or   386
7560         work   365
3326         have   360
3094         from   353
4552         more   346
590          also   331
3856           it   314
1771    countries   312
627            an   303
4729          not   300
798            at   290
3322          has   274
7513        which   273
6935        their   266
6929         than   256
3883          job   248
7222  unemployment   243
1162          can   241
4317       market   235
4921        other   228
7451          was   220
6201       sector   215
6946        these   213
3269       growth   212
6947         they   209
6735         such   209
2396     economic   202
6451       social   199
3601       income   189
7059     training   187
4804         oecd   186
939          been   184
```



Exercise 13

With stop word removal:

```
docs = text_df.text
(top_50_words, plot, pct) = analyze_frequency(docs, 'english')

print('Level of cumulation percentage:', pct)
print()
print('Top 50 words:')
top_50_words
```

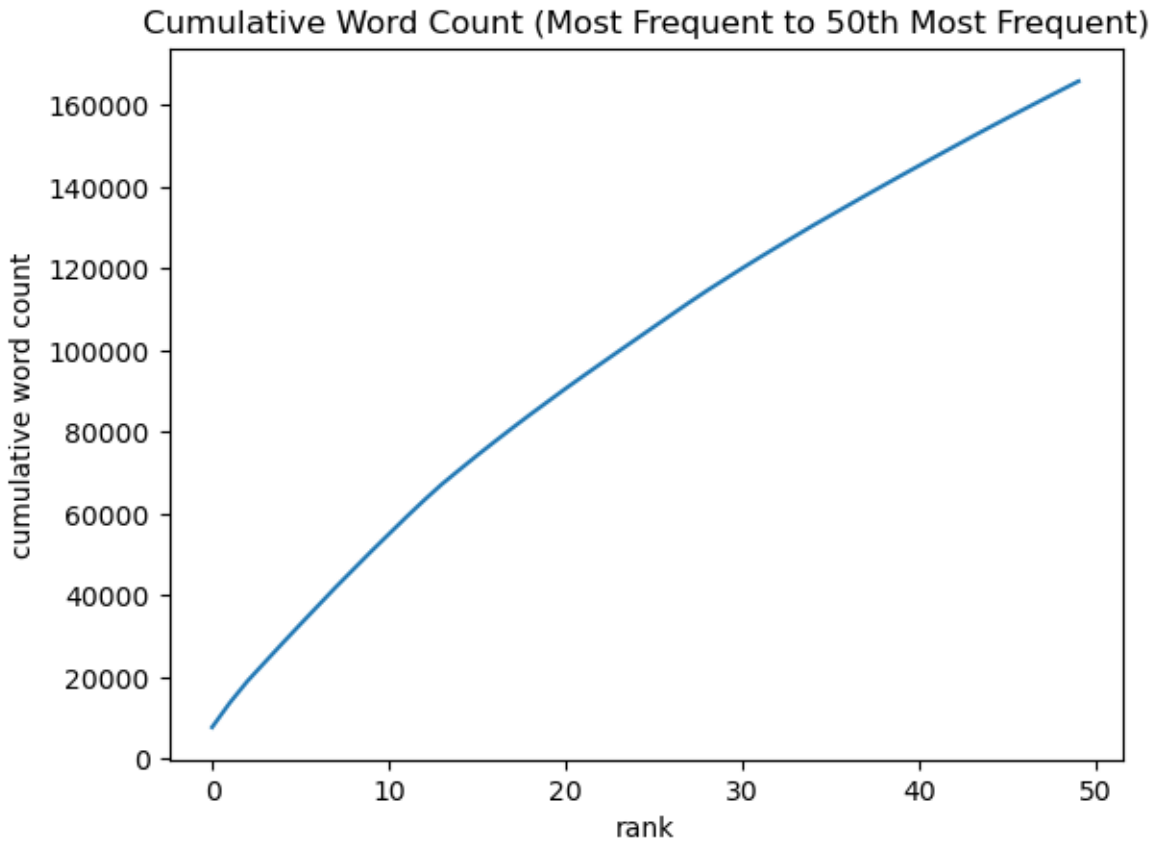
```
Level of cumulation percentage: 12.27%

Top 50 words:
```

```
               term  freq
10295      countries  7761
44859          women  5984
12072    development  5312
19188         health  4685
44337          water  4664
33322         public  4591
38326         social  4538
13847      education  4535
31876         policy  4367
21846  international  4360
```

(continues on next page)

(continued from previous page)

```
24043           law  4240
14495        energy  4224
27973      national  4087
35753        rights  3905
29304          oecd  3547
13726      economic  3517
43320           use  3391
28342           new  3337
24376         level  3267
20913        income  3171
32235       poverty  3167
11084          data  3078
8513         climate  3033
18280     government  3012
7227            care  3002
37356      services  2999
17714        gender  2991
20001         human  2975
5194           based  2879
39986       support  2759
44896          work  2742
19483          high  2674
4016          areas  2611
37030        sector  2583
41272          time  2581
31873      policies  2477
36770        school  2466
25497    management  2465
2098           access  2424
7882           change  2416
18543        growth  2386
19484        higher  2367
21274    information  2363
20898      including  2357
15415        example  2318
24819          local  2285
33571       quality  2255
20724     important  2236
25028           low  2217
12270     different  2211
```



Without stop word removal:

```
(top_50_words, plot, pct) = analyze_frequency(docs, None)

print('Level of cumulation percentage:', pct)
print()
print('Top 50 words:')
top_50_words
```

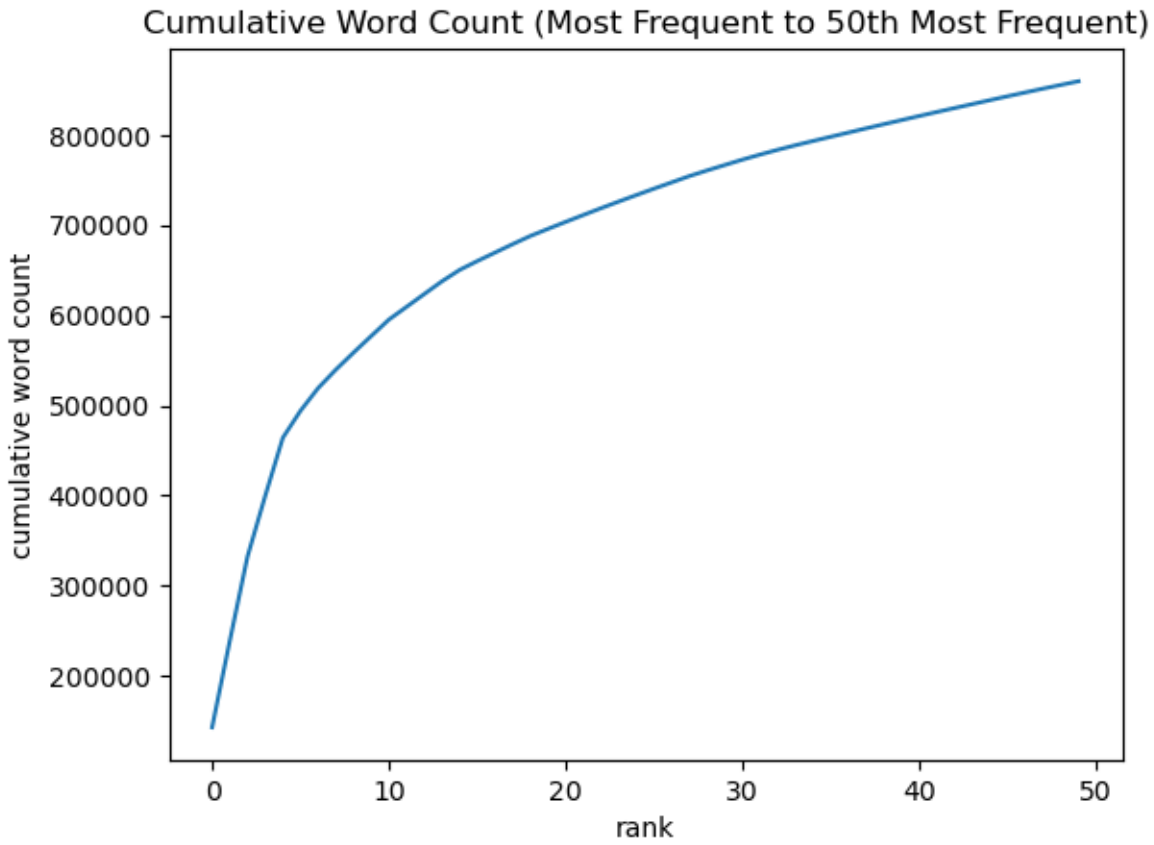
```
Level of cumulation percentage: 36.81%

Top 50 words:
```

	term	freq
41229	the	143100
29487	of	95834
3469	and	93357
20921	in	67152
41630	to	64701
16900	for	30010
22403	is	25175
41221	that	20395
4217	as	18926
29700	on	18365
4040	are	18215
6918	by	14171
45094	with	14162

(continues on next page)

(continued from previous page)

```
41378           this   14153
5337              be   12608
17320           from    9833
29892             or    9382
22504             it    9364
19212           have    9070
3397              an    7885
10354      countries    7761
4447              at    7643
19159            has    7551
41248          their    7356
28988            not    7182
3182            also    7151
44893          which    6968
27490           more    6953
41327          these    6246
7121             can    6016
45152          women    5984
40001           such    5612
12135    development    5312
30114          other    5018
41211           than    4751
19298         health    4685
41339           they    4666
44606          water    4664
33516         public    4591
38540         social    4538
13916      education    4535
5660         between    4466
5401            been    4377
32070         policy    4367
21975  international    4360
44573            was    4339
24180            law    4240
14571         energy    4224
6886              but    4088
28132       national    4087
```



Frequent words that are not stop words, such as 'countries', 'women', and 'development', occur often enough to show up in both lists. As one might expect, stop words such as 'the', 'of', and 'and' occur much more frequently than terms that are not stop words. As a result, the level of cumulation percentage is much smaller and the cumulative word count curve is more linear with stop word removal than without stop word removal.

14.8.3 Document Embedding

Exercise 1

After creating `sentence_df`, we can compare its dimensions to those of `text_df`.

```
def tokenize_into_sentences(corpus):
    corpus_sentence = []
    corpus_sdg = []
    corpus_sample = []
    for (text, sdg, i) in iter(zip(corpus.text, corpus.sdg, corpus.index)):
        sentences = nltk.sent_tokenize(text)
        corpus_sentence += sentences
        corpus_sdg += [sdg]*len(sentences)
        corpus_sample += [i]*len(sentences)
    sentence_df = pd.DataFrame({"text": corpus_sentence, "sdg": corpus_sdg, "sample":␣
↪corpus_sample})
    return sentence_df

sentence_df = tokenize_into_sentences(text_df)
print('text_df dimensions:', text_df.shape)
print('sentence_df dimensions:', sentence_df.shape)
```

```
text_df dimensions: (24669, 7)
sentence_df dimensions: (92839, 3)
```

The dimensions of `text_df` represent the number of sample texts and the number of features, respectively. The dimensions of `sentence_df`represent the number of sentences and the number of features, respectively.

Exercise 2

Student answers may vary. As a sample answer, we choose a text containing direct quotations, which can increase the difficulty of sentence tokenization:

```
text_df.text.loc[465]
```

```
'When asked “Have you no morals?” Alfred Doolittle in George Bernard Shaw’s␣
↪Pygmalion answered: “Can’t afford them governor. Neither could you if you was as␣
↪poor as me.” The modern concept of human rights underpins a moral society and␣
↪holds governments responsible for fulfilling these rights. From informed consent␣
↪to the right to privacy civil and political rights have dominated the human␣
↪rights focus of the HIV-1 epidemic. Yet the economic and social rights of people␣
↪with HIV-1 infection in particular the rights to health care and to share in␣
↪scientific advances are glaringly disparate between rich and poor countries.␣
↪This disparity has become the focus of debate in transnational HIV-1 vaccine␣
↪research. (excerpt)'
```

Despite the increased difficulty, the sentence tokenizer is able to separate out the sentences correctly:

```
sentence_df[sentence_df['sample'] == 465]
```

	text	sdg	sample
1785	When asked “Have you no morals?” Alfred Doolit...	16	465
1786	Neither could you if you was as poor as me.” T...	16	465
1787	From informed consent to the right to privacy ...	16	465
1788	Yet the economic and social rights of people w...	16	465
1789	This disparity has become the focus of debate ...	16	465
1790	(excerpt)	16	465

Exercise 3

Answers may vary depending on the samples chosen. For simplicity, this sample solution chooses two samples in `text_df` with the same number of sentences.

```
samples = text_df.loc[[32,6]]
sentences = tokenize_into_sentences(samples)
sentences
```

	text	sdg	sample
0	This points to the possibility that the effect...	1	32
1	One possible explanation for this is that incr...	1	32
2	These results are similar to those obtained by...	1	32
3	This analysis is presented in the following se...	1	32
4	Prescription rates appear to be higher where l...	8	6
5	There is also a possible relationship between ...	8	6
6	This may arise after the definition of disabil...	8	6
7	Krueger (2017(47)) found that around one-fifth...	8	6

```
# change this to your own embedding directory
embedding_dir = "embeddings/"

# load the embedding
embed = hub.load(embedding_dir + "universal-sentence-encoder_4")
```

```
sdg1_embedding = embed(sentences[sentences['sdg'] == 1].text.tolist())
sdg8_embedding = embed(sentences[sentences['sdg'] == 8].text.tolist())
np.tensordot(sdg1_embedding, sdg8_embedding)
```

```
array(0.2418205, dtype=float32)
```

```
# import libraries
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
import matplotlib as mpl
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.neural_network import MLPClassifier

from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

# change this to your own data directory
data_dir = "data/"

# read and preprocess data
text_file_name = "osdg-community-data-v2023-01-01.csv"
text_df = pd.read_csv(data_dir + text_file_name,sep = "\t",  quotechar='"')
col_names = text_df.columns.values[0].split('\t')
text_df[col_names] = text_df[text_df.columns.values[0]].apply(lambda x: pd.
 ↪Series(str(x).split("\t")))
text_df = text_df.astype({'sdg':int, 'labels_negative': int, 'labels_positive':int,
 ↪'agreement': float}, copy=True)
text_df.drop(text_df.columns.values[0], axis=1, inplace=True)
text_df = text_df.query("agreement > 0.5 and (labels_positive - labels_negative) > 2")
text_df.reset_index(inplace=True, drop=True)
```

```
docs = text_df.text
categories = text_df.sdg
X_train, X_test, y_train, y_test = \
    train_test_split(docs, categories, test_size=0.33, random_state=7)
```

14.9 Solutions to Exercises: Sections 5 to 7

14.9.1 Classification of Texts

For exercises 1 to 3, the below code will be used to establish the dataset used for all next classifications.

```
X_train_tfidf_vectorizer = TfidfVectorizer(ngram_range=(1,1), stop_words = "english",␣
 ↪min_df=5)
X_train_tfidf_vectorizer.fit(X_train)
X_train_tfidf_vector = X_train_tfidf_vectorizer.transform(X_train)
X_test_tfidf_vector = X_train_tfidf_vectorizer.transform(X_test)
```

Exercise 1

```
tfidf_mlp_clf = MLPClassifier(random_state=1, max_iter=100).fit(X_train_tfidf_vector,␣
 ↪y_train)
y_pred = tfidf_mlp_clf.predict(X_test_tfidf_vector)
tfidf_mlp_clf.score(X_test_tfidf_vector, y_test)
print(metrics.classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           1       0.79      0.81      0.80       481
           2       0.82      0.87      0.84       316
           3       0.94      0.93      0.94       674
           4       0.92      0.94      0.93       863
           5       0.92      0.92      0.92       920
           6       0.90      0.91      0.91       465
           7       0.88      0.88      0.88       730
           8       0.68      0.61      0.64       353
           9       0.79      0.80      0.79       328
          10       0.64      0.58      0.61       256
          11       0.85      0.88      0.86       462
          12       0.85      0.78      0.81       217
          13       0.85      0.87      0.86       443
          14       0.93      0.92      0.93       263
          15       0.89      0.85      0.87       313
          16       0.97      0.97      0.97      1057

    accuracy                           0.88      8141
   macro avg       0.85      0.85      0.85      8141
weighted avg       0.88      0.88      0.88      8141
```

Exercise 2

```
tfidf_multinomialNB_clf = MultinomialNB().fit(X_train_tfidf_vector, y_train)
y_pred = tfidf_multinomialNB_clf.predict(X_test_tfidf_vector)
print(metrics.classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           1       0.70      0.73      0.71       481
           2       0.90      0.67      0.77       316
           3       0.92      0.90      0.91       674
           4       0.76      0.96      0.85       863
           5       0.62      0.94      0.75       920
```

(continues on next page)

(continued from previous page)

```
           6       0.86      0.81      0.84       465
           7       0.60      0.97      0.74       730
           8       0.85      0.08      0.15       353
           9       0.91      0.32      0.48       328
          10       0.89      0.12      0.21       256
          11       0.86      0.75      0.80       462
          12       0.97      0.30      0.46       217
          13       0.84      0.80      0.82       443
          14       0.96      0.61      0.75       263
          15       0.96      0.64      0.77       313
          16       0.86      0.98      0.91      1057

    accuracy                           0.77      8141
   macro avg       0.84      0.66      0.68      8141
weighted avg       0.81      0.77      0.74      8141
```

Exercise 3

```
from sklearn.linear_model import RidgeClassifier

tfidf_ridge_clf = RidgeClassifier(tol=1e-2, solver="sparse_cg")
tfidf_ridge_clf = tfidf_ridge_clf.fit(X_train_tfidf_vector, y_train)
y_pred = tfidf_ridge_clf.predict(X_test_tfidf_vector)
print(metrics.classification_report(y_test,y_pred, digits = 4))
```

```
              precision    recall  f1-score   support

           1     0.8174    0.8004    0.8088       481
           2     0.8192    0.8892    0.8528       316
           3     0.9294    0.9377    0.9335       674
           4     0.9109    0.9594    0.9345       863
           5     0.9092    0.9467    0.9276       920
           6     0.9188    0.9247    0.9218       465
           7     0.8714    0.9096    0.8901       730
           8     0.7186    0.6006    0.6543       353
           9     0.8233    0.7957    0.8093       328
          10     0.6990    0.5625    0.6234       256
          11     0.8690    0.8615    0.8652       462
          12     0.8836    0.7696    0.8227       217
          13     0.8521    0.8713    0.8616       443
          14     0.9423    0.9316    0.9369       263
          15     0.9112    0.8850    0.8979       313
          16     0.9513    0.9612    0.9562      1057

    accuracy                         0.8840      8141
   macro avg     0.8642    0.8504    0.8560      8141
weighted avg     0.8815    0.8840    0.8820      8141
```

Exercise 4

Implementations may vary. One example is given below:

```
def ClassifyDocs(data, classifier_algorithm, params):
    docs = data.text
    categories = data.sdg
    X_train, X_test, y_train, y_test = train_test_split(docs, categories, test_size=0.
 ↪33, random_state=7)
```

(continues on next page)

(continued from previous page)

```
    X_train_tfidf_vectorizer = TfidfVectorizer(ngram_range=(1,1), stop_words =
↪"english", min_df=5)
    X_train_tfidf_vectorizer.fit(X_train)
    X_train_tfidf_vector = X_train_tfidf_vectorizer.transform(X_train)
    X_test_tfidf_vector = X_train_tfidf_vectorizer.transform(X_test)

    if classifier_algorithm == "mlp":
        tfidf_mlp_clf = MLPClassifier(random_state=params[0], max_iter=params[1]).
↪fit(X_train_tfidf_vector, y_train)
        y_pred = tfidf_mlp_clf.predict(X_test_tfidf_vector)
        tfidf_mlp_clf.score(X_test_tfidf_vector, y_test)
        print(metrics.classification_report(y_test,y_pred))

    elif classifier_algorithm == "multinomialNB":
        tfidf_multinomialNB_clf = MultinomialNB().fit(X_train_tfidf_vector, y_train)
        y_pred = tfidf_multinomialNB_clf.predict(X_test_tfidf_vector)
        print(metrics.classification_report(y_test,y_pred))

    elif classifier_algorithm == "ridge":
        tfidf_ridge_clf = RidgeClassifier(tol=params[0], solver=params[1])
        tfidf_ridge_clf = tfidf_ridge_clf.fit(X_train_tfidf_vector, y_train)
        y_pred = tfidf_ridge_clf.predict(X_test_tfidf_vector)
        print(metrics.classification_report(y_test,y_pred, digits = 4))

    else:
        print("Invalid classifier algorithm")
```

Exercise 5

```
print("MLP")
ClassifyDocs(text_df, "mlp", [1,100])
print("MultinomialNB")
ClassifyDocs(text_df, "multinomialNB", [])
print("Ridge")
class_params = [1e-2, "sparse_cg"]
ClassifyDocs(text_df, "ridge", class_params)
```

```
MLP
              precision    recall  f1-score   support

           1       0.79      0.81      0.80       481
           2       0.82      0.87      0.84       316
           3       0.94      0.93      0.94       674
           4       0.92      0.94      0.93       863
           5       0.92      0.92      0.92       920
           6       0.90      0.91      0.91       465
           7       0.88      0.88      0.88       730
           8       0.68      0.61      0.64       353
           9       0.79      0.80      0.79       328
          10       0.64      0.58      0.61       256
          11       0.85      0.88      0.86       462
          12       0.85      0.78      0.81       217
          13       0.85      0.87      0.86       443
          14       0.93      0.92      0.93       263
          15       0.89      0.85      0.87       313
```

(continues on next page)

(continued from previous page)

```
          16       0.97      0.97      0.97      1057

    accuracy                           0.88      8141
   macro avg       0.85      0.85      0.85      8141
weighted avg       0.88      0.88      0.88      8141

MultinomialNB
              precision    recall  f1-score   support

           1       0.70      0.73      0.71       481
           2       0.90      0.67      0.77       316
           3       0.92      0.90      0.91       674
           4       0.76      0.96      0.85       863
           5       0.62      0.94      0.75       920
           6       0.86      0.81      0.84       465
           7       0.60      0.97      0.74       730
           8       0.85      0.08      0.15       353
           9       0.91      0.32      0.48       328
          10       0.89      0.12      0.21       256
          11       0.86      0.75      0.80       462
          12       0.97      0.30      0.46       217
          13       0.84      0.80      0.82       443
          14       0.96      0.61      0.75       263
          15       0.96      0.64      0.77       313
          16       0.86      0.98      0.91      1057

    accuracy                           0.77      8141
   macro avg       0.84      0.66      0.68      8141
weighted avg       0.81      0.77      0.74      8141

Ridge
              precision    recall  f1-score   support

           1     0.8174    0.8004    0.8088       481
           2     0.8192    0.8892    0.8528       316
           3     0.9294    0.9377    0.9335       674
           4     0.9109    0.9594    0.9345       863
           5     0.9092    0.9467    0.9276       920
           6     0.9188    0.9247    0.9218       465
           7     0.8714    0.9096    0.8901       730
           8     0.7186    0.6006    0.6543       353
           9     0.8233    0.7957    0.8093       328
          10     0.6990    0.5625    0.6234       256
          11     0.8690    0.8615    0.8652       462
          12     0.8836    0.7696    0.8227       217
          13     0.8521    0.8713    0.8616       443
          14     0.9423    0.9316    0.9369       263
          15     0.9112    0.8850    0.8979       313
          16     0.9513    0.9612    0.9562      1057

    accuracy                           0.8840      8141
   macro avg     0.8642    0.8504    0.8560      8141
weighted avg     0.8815    0.8840    0.8820      8141
```

14.9.2 Evaluating Classification

Exercise 1

First, we do the prep needed before making the ROC curve:

```
from sklearn import preprocessing
from sklearn.multiclass import OneVsRestClassifier
from sklearn.preprocessing import LabelBinarizer

import tensorflow as tf
import tensorflow_hub as hub

# change this to your own embedding directory
embedding_dir = ""

# load the embedding
embed = hub.load(embedding_dir + "universal-sentence-encoder_4")

text_df["embedding"] = list(embed(text_df.text))

file_name = "sdg_names_definitions.csv"
sdg_names = pd.read_csv(data_dir + file_name)

docs = text_df.embedding.tolist()
scaler = preprocessing.MinMaxScaler().fit(docs)
X = scaler.transform(docs)
y = text_df.sdg

label_binarizer = LabelBinarizer().fit(y)
y_onehot = label_binarizer.transform(y)
n_classes = len(label_binarizer.classes_)
class_names = [sdg_names[sdg_names["sdg"] == label_binarizer.classes_[i]].sdg_name.
 ↪item() \
               for i in range(n_classes)]
```

```
2024-07-12 16:12:25.149972: I tensorflow/core/common_runtime/executor.cc:1197] [/
 ↪device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an␣
 ↪error and you can ignore this message): INVALID_ARGUMENT: You must feed a value␣
 ↪for placeholder tensor 'inputs' with dtype string
         [[{{node inputs}}]]
```

Then, we can make the curve itself. First for the Naive Bayes classification:

```
X_train, X_test, y_train, y_test = train_test_split(X, y_onehot, test_size=.33,␣
 ↪random_state=0)
ovr_nb_clf = OneVsRestClassifier(MultinomialNB()).fit(X_train,y_train)
y_score_nb = ovr_nb_clf.predict_proba(X_test)
```

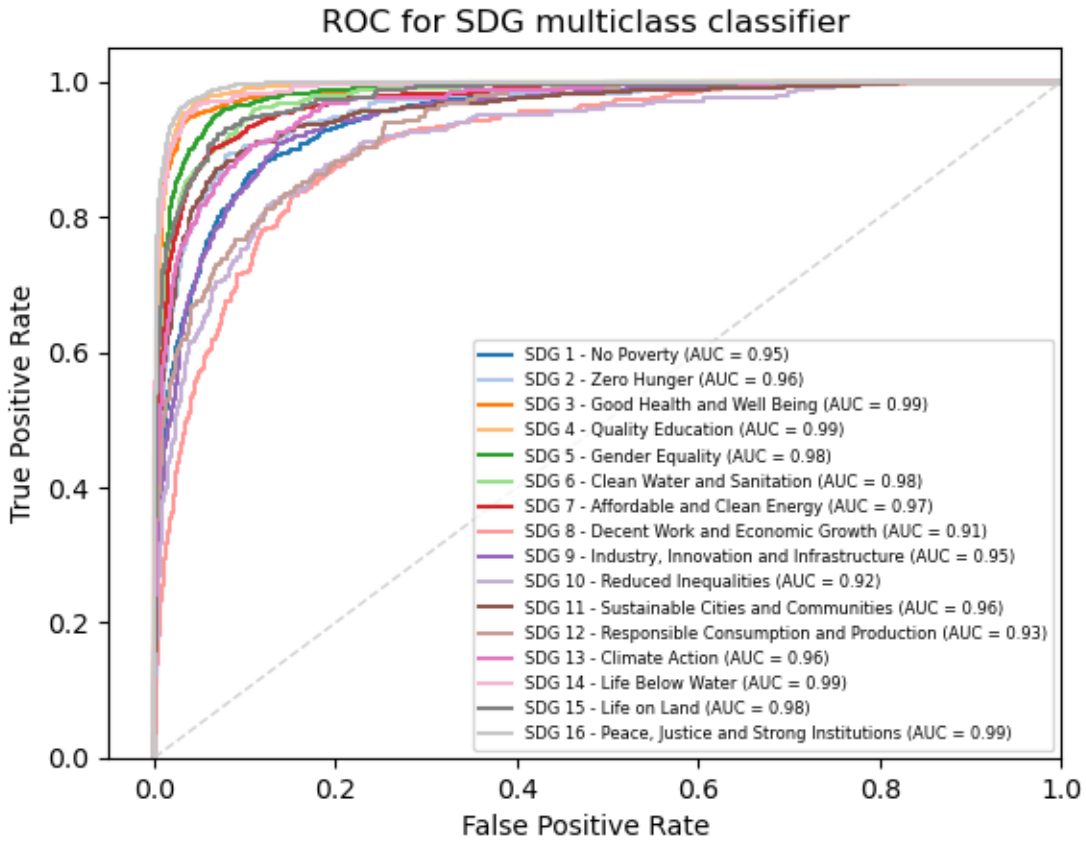
```
from matplotlib import cm
colors = list(cm.get_cmap('tab20').colors)

fpr = dict()
tpr = dict()
roc_auc = dict()
for class_id in range(n_classes):
    fpr[class_id], tpr[class_id], _ = metrics.roc_curve(y_test[:, class_id], y_score_
```

(continues on next page)

(continued from previous page)

```
 ↪nb[:, class_id]) # roc_curve works on binary
    roc_auc[class_id] = metrics.auc(fpr[class_id], tpr[class_id])

for class_id, color in zip(range(n_classes), colors):
    plt.plot(fpr[class_id], tpr[class_id], color=color, lw=1.5,alpha = 1,
             label='SDG {0} - {1} (AUC = {2:0.2f})'
             ''.format(class_id+1, class_names[class_id], roc_auc[class_id]))
plt.plot([0, 1], [0, 1], '--', lw=1, color="lightgrey")
plt.xlim([-0.05, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC for SDG multiclass classifier')
plt.legend(loc="lower right", fontsize=6)
plt.show()
```

```
/var/folders/px/b7vc3nh913zb_m0x36ncftj00000gn/T/ipykernel_86574/1881487763.py:2:␣
 ↪MatplotlibDeprecationWarning: The get_cmap function was deprecated in Matplotlib␣
 ↪3.7 and will be removed two minor releases later. Use ``matplotlib.
 ↪colormaps[name]`` or ``matplotlib.colormaps.get_cmap(obj)`` instead.
  colors = list(cm.get_cmap('tab20').colors)
```


ROC for SDG multiclass classifier
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
SDG 1 - No Poverty (AUC = 0.95)
SDG 2 - Zero Hunger (AUC = 0.96)
SDG 3 - Good Health and Well Being (AUC = 0.99)
SDG 4 - Quality Education (AUC = 0.99)
SDG 5 - Gender Equality (AUC = 0.98)
SDG 6 - Clean Water and Sanitation (AUC = 0.98)
SDG 7 - Affordable and Clean Energy (AUC = 0.97)
SDG 8 - Decent Work and Economic Growth (AUC = 0.91)
SDG 9 - Industry, Innovation and Infrastructure (AUC = 0.95)
SDG 10 - Reduced Inequalities (AUC = 0.92)
SDG 11 - Sustainable Cities and Communities (AUC = 0.96)
SDG 12 - Responsible Consumption and Production (AUC = 0.93)
SDG 13 - Climate Action (AUC = 0.96)
SDG 14 - Life Below Water (AUC = 0.99)
SDG 15 - Life on Land (AUC = 0.98)
SDG 16 - Peace, Justice and Strong Institutions (AUC = 0.99)

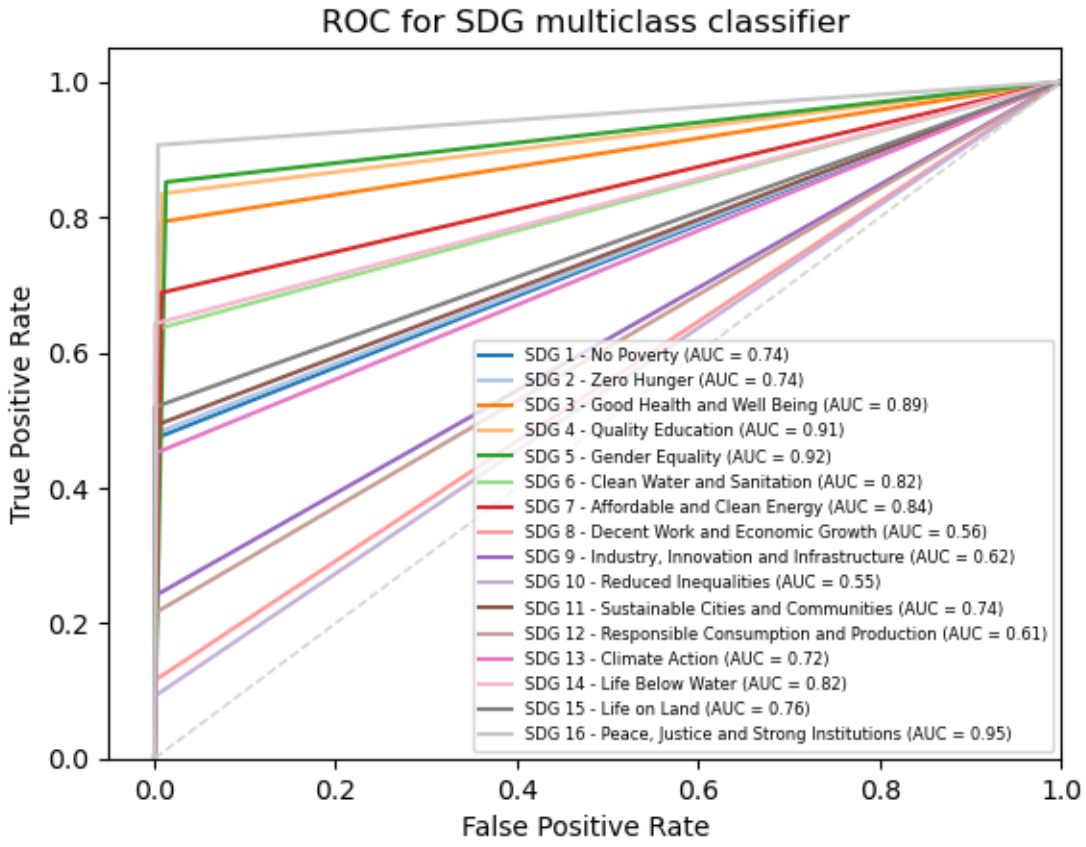
And then for the Ridge classification:

```
ovr_ridge_clf = OneVsRestClassifier(RidgeClassifier(tol=1e-2, solver="sparse_cg")).
 ↪fit(X_train,y_train)
```

(continues on next page)

(continued from previous page)

```
y_score_ridge = ovr_ridge_clf.predict(X_test)

fpr = dict()
tpr = dict()
roc_auc = dict()
for class_id in range(n_classes):
    fpr[class_id], tpr[class_id], _ = metrics.roc_curve(y_test[:, class_id], y_score_
↪ridge[:, class_id]) # roc_curve works on binary
    roc_auc[class_id] = metrics.auc(fpr[class_id], tpr[class_id])

for class_id, color in zip(range(n_classes), colors):
    plt.plot(fpr[class_id], tpr[class_id], color=color, lw=1.5,alpha = 1,
             label='SDG {0} - {1} (AUC = {2:0.2f})'
             ''.format(class_id+1, class_names[class_id], roc_auc[class_id]))
plt.plot([0, 1], [0, 1], '--', lw=1, color="lightgrey")
plt.xlim([-0.05, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC for SDG multiclass classifier')
plt.legend(loc="lower right", fontsize=6)
plt.show()
```


ROC for SDG multiclass classifier
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
SDG 1 - No Poverty (AUC = 0.74)
SDG 2 - Zero Hunger (AUC = 0.74)
SDG 3 - Good Health and Well Being (AUC = 0.89)
SDG 4 - Quality Education (AUC = 0.91)
SDG 5 - Gender Equality (AUC = 0.92)
SDG 6 - Clean Water and Sanitation (AUC = 0.82)
SDG 7 - Affordable and Clean Energy (AUC = 0.84)
SDG 8 - Decent Work and Economic Growth (AUC = 0.56)
SDG 9 - Industry, Innovation and Infrastructure (AUC = 0.62)
SDG 10 - Reduced Inequalities (AUC = 0.55)
SDG 11 - Sustainable Cities and Communities (AUC = 0.74)
SDG 12 - Responsible Consumption and Production (AUC = 0.61)
SDG 13 - Climate Action (AUC = 0.72)
SDG 14 - Life Below Water (AUC = 0.82)
SDG 15 - Life on Land (AUC = 0.76)
SDG 16 - Peace, Justice and Strong Institutions (AUC = 0.95)

When students are making the plot for ridge regression, they will likely notice that ridge classifiers do not have a `predict_proba` method. Feel free to encourage students to look to other resources for overcoming this problem. Alternatively, ask them why they might think this is the case - the reason for this is because ridge regression does not calculate probabilities for its results in its basic form but must rather interpret the assigned values from the regression into some

sort of probability, which it does not do by default.

Exercise 2

Answers may vary depending on the chosen goals but should follow the example given in the chapter.

Exercise 3

First, we perform the classification:

```
docs = text_df.embedding.tolist()
scaler = preprocessing.MinMaxScaler().fit(docs)
X = scaler.transform(docs)
y = text_df.sdg

label_binarizer = LabelBinarizer().fit(y)
y_onehot = label_binarizer.transform(y)
n_classes = len(label_binarizer.classes_)
class_names = [sdg_names[sdg_names["sdg"] == label_binarizer.classes_[i]].sdg_name.
 ↪item() \
               for i in range(n_classes)]

X_train, X_test, y_train, y_test = train_test_split(X, y_onehot, test_size=.33,␣
 ↪random_state=0)
ovr_nb_clf = OneVsRestClassifier(MultinomialNB()).fit(X_train,y_train)
y_score_nb = ovr_nb_clf.predict_proba(X_test)
```

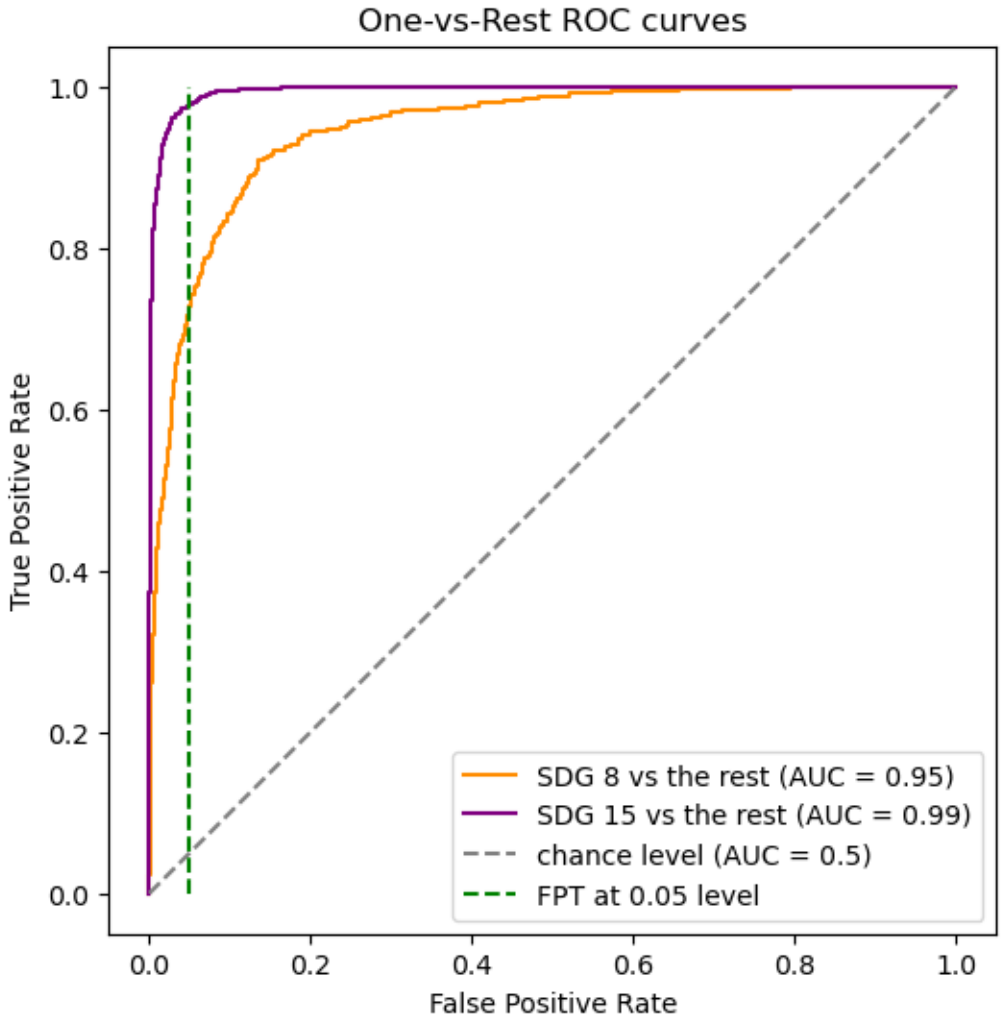
Then we can make the plot, first for our Naive Bayes Classification. Note that we did SDG 15 for this plot as SDG yielded a plotting error for us; students can change goals as well.

```
from sklearn.metrics import RocCurveDisplay

label_binarizer = LabelBinarizer().fit(y_train)
y_onehot_test = label_binarizer.transform(y_test)

fig, ax = plt.subplots(figsize=(6,6))

class_of_interest = 8 # SDG 8
class_id = np.flatnonzero(label_binarizer.classes_ == class_of_interest)[0]
RocCurveDisplay.from_predictions(
    y_onehot_test[:, class_id],
    y_score_nb[:, class_id],
    name=f"SDG {class_of_interest} vs the rest",
    color="darkorange",
    ax = ax,
)

class_of_interest = 15
class_id = np.flatnonzero(label_binarizer.classes_ == class_of_interest)[0]
RocCurveDisplay.from_predictions(
    y_onehot_test[:, class_id],
    y_score_nb[:, class_id],
    name=f"SDG {class_of_interest} vs the rest",
    color="purple",
    ax = ax,
)

plt.plot([0, 1], [0, 1], "--", label="chance level (AUC = 0.5)", color = "grey")
plt.plot([0.05, 0.05], [0, 1], "--", label="FPT at 0.05 level", color = "green")
```

(continues on next page)

(continued from previous page)

```
plt.axis("square")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("One-vs-Rest ROC curves")
plt.legend()
plt.show()
```



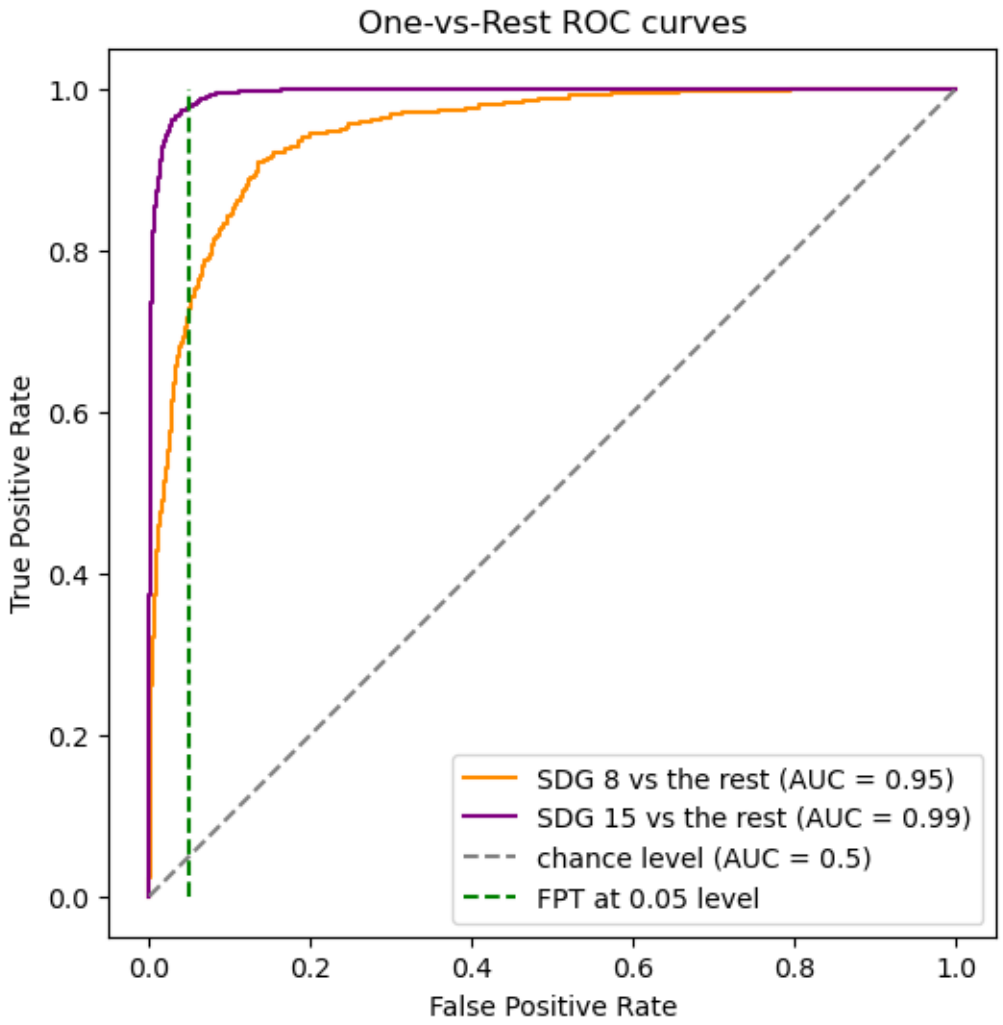
And then the Ridge classifier:

```
X_train, X_test, y_train, y_test = train_test_split(X, y_onehot, test_size=.33,␣
 ↪random_state=0)
ovr_ridge_clf = OneVsRestClassifier(RidgeClassifier(tol=1e-2, solver="sparse_cg")).
 ↪fit(X_train,y_train)
y_score_ridge = ovr_nb_clf.predict(X_test)

fig, ax = plt.subplots(figsize=(6,6))

class_of_interest = 8 # SDG 8
class_id = np.flatnonzero(label_binarizer.classes_ == class_of_interest)[0]
```

(continues on next page)

(continued from previous page)

```
RocCurveDisplay.from_predictions(
    y_onehot_test[:, class_id],
    y_score_nb[:, class_id],
    name=f"SDG {class_of_interest} vs the rest",
    color="darkorange",
    ax = ax,
)

class_of_interest = 15
class_id = np.flatnonzero(label_binarizer.classes_ == class_of_interest)[0]
RocCurveDisplay.from_predictions(
    y_onehot_test[:, class_id],
    y_score_nb[:, class_id],
    name=f"SDG {class_of_interest} vs the rest",
    color="purple",
    ax = ax,
)

plt.plot([0, 1], [0, 1], "--", label="chance level (AUC = 0.5)", color = "grey")
plt.plot([0.05, 0.05], [0, 1], "--", label="FPT at 0.05 level", color = "green")

plt.axis("square")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("One-vs-Rest ROC curves")
plt.legend()
plt.show()
```



Exercise 4

Implementations may vary. One example is given below:

```
def ClassifyDocs_Advanced(data, classifier_algorithm, params, roc_plot = False, quick_
↪metrics = False):
    docs = data.text
    categories = data.sdg
    X_train, X_test, y_train, y_test = train_test_split(docs, categories, test_size=0.
↪33, random_state=7)

    X_train_tfidf_vectorizer = TfidfVectorizer(ngram_range=(1,1), stop_words =
↪"english", min_df=5)
    X_train_tfidf_vectorizer.fit(X_train)
    X_train_tfidf_vector = X_train_tfidf_vectorizer.transform(X_train)
    X_test_tfidf_vector = X_train_tfidf_vectorizer.transform(X_test)

    metrics_report = ""

    if classifier_algorithm == "mlp":
        tfidf_mlp_clf = MLPClassifier(random_state=params[0], max_iter=params[1]).
↪fit(X_train_tfidf_vector, y_train)
```

(continues on next page)

(continued from previous page)

```
        y_pred = tfidf_mlp_clf.predict(X_test_tfidf_vector)
        tfidf_mlp_clf.score(X_test_tfidf_vector, y_test)
        metrics_report = metrics.classification_report(y_test,y_pred)

    elif classifier_algorithm == "multinomialNB":
        tfidf_multinomialNB_clf = MultinomialNB().fit(X_train_tfidf_vector, y_train)
        y_pred = tfidf_multinomialNB_clf.predict(X_test_tfidf_vector)
        metrics_report = metrics.classification_report(y_test,y_pred)

    elif classifier_algorithm == "ridge":
        tfidf_ridge_clf = RidgeClassifier(tol=params[0], solver=params[1])
        tfidf_ridge_clf = tfidf_ridge_clf.fit(X_train_tfidf_vector, y_train)
        y_pred = tfidf_ridge_clf.predict(X_test_tfidf_vector)
        metrics_report = metrics.classification_report(y_test,y_pred)

    else:
        print("Invalid classifier algorithm")
        return 0

    if quick_metrics:
        #print only accuracy, precision, recall, f1-score from metrics_report
        print(metrics_report.split("\n")[0])
    else:
        print(metrics_report)

    if roc_plot:
        #plot ROC curve
        label_binarizer = LabelBinarizer().fit(y_test)
        y_onehot_test = label_binarizer.transform(y_test)
        y_score = None
        if classifier_algorithm == "mlp":
            y_score = tfidf_mlp_clf.predict_proba(X_test_tfidf_vector)
        elif classifier_algorithm == "multinomialNB":
            y_score = tfidf_multinomialNB_clf.predict_proba(X_test_tfidf_vector)
        elif classifier_algorithm == "ridge":
            y_score = tfidf_ridge_clf.decision_function(X_test_tfidf_vector)
        else:
            print("Invalid classifier algorithm")
            return 0

        fig, ax = plt.subplots(figsize=(6,6))

        from matplotlib import cm
        colors = list(cm.get_cmap('tab20').colors)

        fpr = dict()
        tpr = dict()
        roc_auc = dict()
        for class_id in range(n_classes):
            fpr[class_id], tpr[class_id], _ = metrics.roc_curve(y_test[:, class_id],␣
↪y_score_nb[:, class_id]) # roc_curve works on binary
            roc_auc[class_id] = metrics.auc(fpr[class_id], tpr[class_id])

        for class_id, color in zip(range(n_classes), colors):
            plt.plot(fpr[class_id], tpr[class_id], color=color, lw=1.5,alpha = 1,
             label='SDG {0} - {1} (AUC = {2:0.2f})'
             ''.format(class_id+1, class_names[class_id], roc_auc[class_id]))
```

(continues on next page)

(continued from previous page)

```
        plt.plot([0, 1], [0, 1], '--', lw=1, color="lightgrey")
        plt.xlim([-0.05, 1.0])
        plt.ylim([0.0, 1.05])
        plt.xlabel('False Positive Rate')
        plt.ylabel('True Positive Rate')
        plt.title('ROC for SDG multiclass classifier')
        plt.legend(loc="lower right", fontsize=6)
        plt.show()
```

Exercise 5

A point at (1,1) corresponds to a model that predicts everything as a positive, leading to all data points being labeled as positive (so a 100% true and false positive rate). A point at (0,0) corresponds to a model that predicts everything as a negative (so a 0% true and false positive rate). While students do not have to elaborate on this, this hopefully demonstrates that a model with 100% true positive rate or a model with 0% false positive rate is not necessarily a good thing.

14.9.3 Applications

Exercise 1

Since we are classifying new text data, we do not need to use as many data points for testing purposes. In previous chapters, more testing points are better for model evaluation, while more training data is better to make a more accurate model. In this case, having more training data can make for a more classification of the new texts.

Exercise 2

For these exercises, the training process can take a while, but it is interruptable, as in the solutions given below. The results for the MLP classifier are given as follows:

```
import requests
from bs4 import BeautifulSoup
url = "https://news.un.org/en/story/2023/07/1138767"
response = requests.get(url)
soup = BeautifulSoup(response.content, "html.parser")

from nltk import sent_tokenize
page_sentences = []
for tag in soup.find_all('p'):
        if (len(tag.text)>=30):
            sentences = sent_tokenize(tag.text)
            page_sentences = page_sentences + sentences
page_sentence_df = pd.DataFrame({"text": page_sentences})

# change this to your own data directory
data_dir = "data/"

# read and preprocess data
text_file_name = "osdg-community-data-v2023-01-01.csv"
text_df = pd.read_csv(data_dir + text_file_name,sep = "\t",  quotechar='"')
col_names = text_df.columns.values[0].split('\t')
text_df[col_names] = text_df[text_df.columns.values[0]].apply(lambda x: pd.
 ↪Series(str(x).split("\t")))
text_df = text_df.astype({'sdg':int, 'labels_negative': int, 'labels_positive':int,
 ↪'agreement': float}, copy=True)
text_df.drop(text_df.columns.values[0], axis=1, inplace=True)
text_df = text_df.query("agreement > 0.5 and (labels_positive - labels_negative) > 2")
```

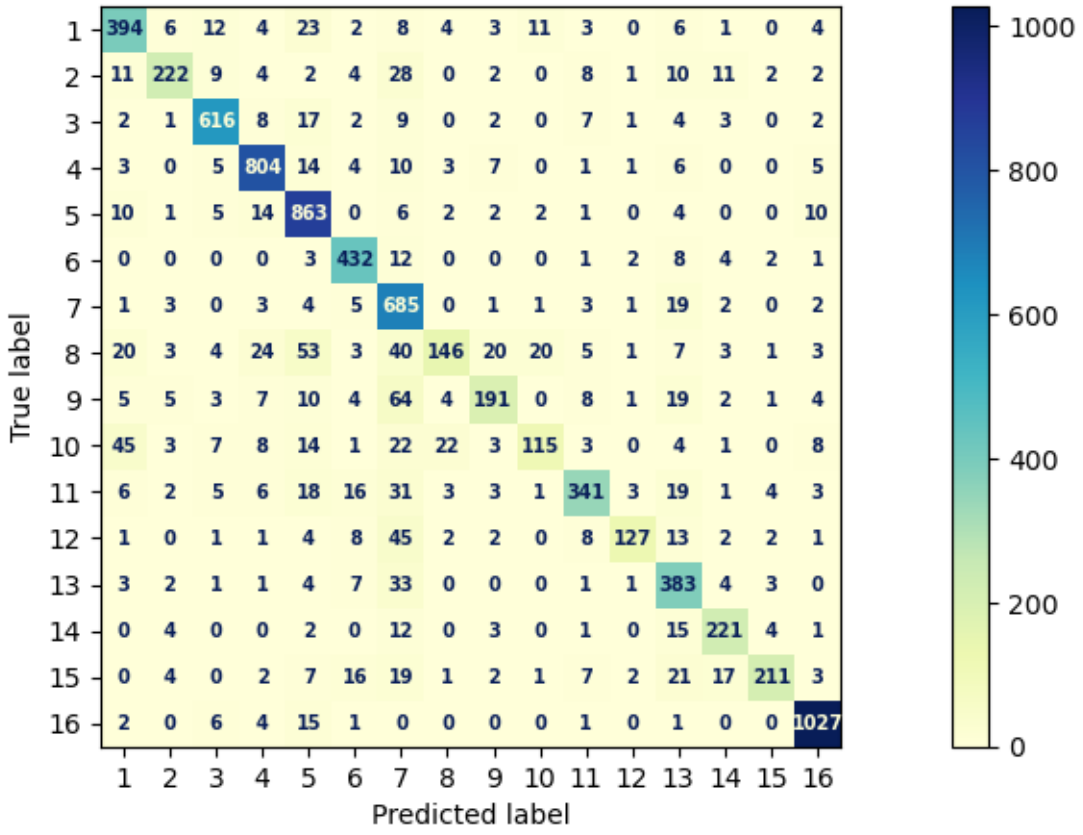
(continues on next page)

(continued from previous page)

```
text_df.reset_index(inplace=True, drop=True)

docs = text_df.text
categories = text_df.sdg
X_train, X_test, y_train, y_test = \
    train_test_split(docs, categories, test_size=0.33, random_state=7)

X_train_count_vectorizer = CountVectorizer(ngram_range=(2,2), stop_words = "english" )
X_train_count_vectorizer.fit(X_train)
X_train_count_vector = X_train_count_vectorizer.transform(X_train)
X_test_count_vector = X_train_count_vectorizer.transform(X_test)

count_MLP_clf = MLPClassifier(random_state=1, max_iter=100).fit(X_train_count_vector,␣
 ↪y_train)
y_pred = count_MLP_clf.predict(X_test_count_vector)

fig, ax = plt.subplots(figsize=(15, 5))
font = {'family': 'sans-serif', 'weight': 'heavy','size': 7,} # set font for␣
 ↪displaying confusion matrix
cm = mpl.colormaps["YlGnBu"] # set the color map for displaying confusion matrix
ConfusionMatrixDisplay.from_predictions(y_test, y_pred, text_kw=font, ax=ax, cmap=cm)
print(metrics.classification_report(y_test,y_pred, digits = 4))
```

```
/opt/anaconda3/envs/nlp2/lib/python3.9/site-packages/sklearn/neural_network/_
 ↪multilayer_perceptron.py:693: UserWarning: Training interrupted by user.
  warnings.warn("Training interrupted by user.")
```

```
              precision    recall  f1-score   support

           1     0.7833    0.8191    0.8008       481
           2     0.8672    0.7025    0.7762       316
           3     0.9139    0.9139    0.9139       674
           4     0.9034    0.9316    0.9173       863
           5     0.8196    0.9380    0.8748       920
           6     0.8554    0.9290    0.8907       465
           7     0.6689    0.9384    0.7811       730
           8     0.7807    0.4136    0.5407       353
           9     0.7925    0.5823    0.6714       328
          10     0.7616    0.4492    0.5651       256
          11     0.8546    0.7381    0.7921       462
          12     0.9007    0.5853    0.7095       217
          13     0.7106    0.8646    0.7800       443
          14     0.8125    0.8403    0.8262       263
          15     0.9174    0.6741    0.7772       313
          16     0.9545    0.9716    0.9630      1057

    accuracy                         0.8326      8141
   macro avg     0.8311    0.7682    0.7863      8141
weighted avg     0.8392    0.8326    0.8263      8141
```



Similarly, the solutions for Ridge Regression work as follows:

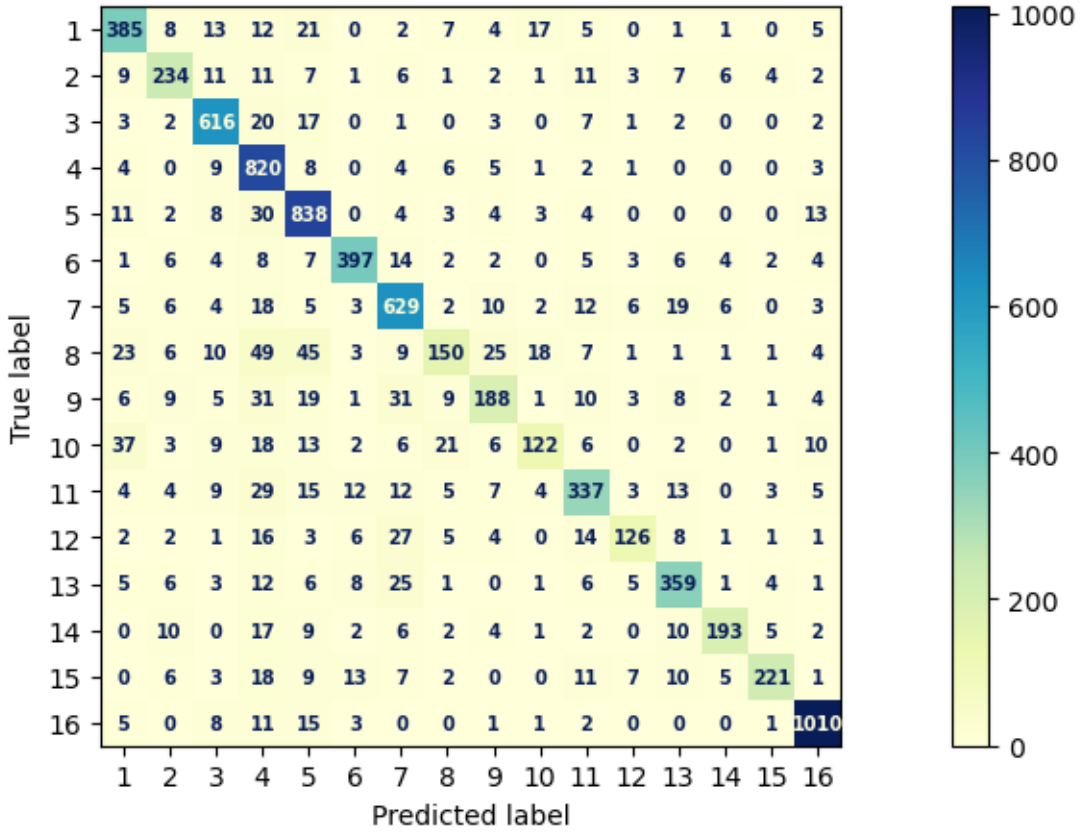
```
count_ridge_clf = RidgeClassifier(tol=1e-2, solver="sparse_cg").fit(X_train_count_
↪vector, y_train)
y_pred = count_ridge_clf.predict(X_test_count_vector)

fig, ax = plt.subplots(figsize=(15, 5))
font = {'family': 'sans-serif', 'weight': 'heavy','size': 7,} # set font for␣
↪displaying confusion matrix
cm = mpl.colormaps["YlGnBu"] # set the color map for displaying confusion matrix
ConfusionMatrixDisplay.from_predictions(y_test, y_pred, text_kw=font, ax=ax, cmap=cm)
print(metrics.classification_report(y_test,y_pred, digits = 4))
```

```
              precision    recall  f1-score   support

           1     0.7700    0.8004    0.7849       481
           2     0.7697    0.7405    0.7548       316
           3     0.8640    0.9139    0.8882       674
           4     0.7321    0.9502    0.8270       863
           5     0.8081    0.9109    0.8564       920
           6     0.8803    0.8538    0.8668       465
           7     0.8033    0.8616    0.8315       730
           8     0.6944    0.4249    0.5272       353
           9     0.7094    0.5732    0.6341       328
          10     0.7093    0.4766    0.5701       256
          11     0.7642    0.7294    0.7464       462
          12     0.7925    0.5806    0.6702       217
          13     0.8049    0.8104    0.8076       443
```

(continues on next page)

(continued from previous page)

```
           14      0.8773    0.7338    0.7992       263
           15      0.9057    0.7061    0.7935       313
           16      0.9439    0.9555    0.9497      1057

     accuracy                         0.8138      8141
    macro avg      0.8018    0.7514    0.7692      8141
 weighted avg      0.8132    0.8138    0.8079      8141
```



Answers comparing these models may vary, but they are largely similar in their results.

Exercise 3

Answers may vary as to how the students classify the website by reading it; however, goal 8 (Decent work and economic growth) or goal 1 (No poverty) should make the most sense.

To classify this document with Naive Bayes, we do the following:

```
url = "https://www.rockefellerfoundation.org/our-work/economic-equity/"
response = requests.get(url)
soup = BeautifulSoup(response.content, "html.parser")

page_sentences = []
for tag in soup.find_all('p'):
        if (len(tag.text)>=30):
            sentences = sent_tokenize(tag.text)
            page_sentences = page_sentences + sentences
page_sentence_df = pd.DataFrame({"text": page_sentences})
```

(continues on next page)

(continued from previous page)

```
count_NB_clf = MultinomialNB().fit(X_train_count_vector, y_train)
y_pred = count_NB_clf.predict(X_test_count_vector)

page_count_vector = X_train_count_vectorizer.transform(pd.Series(page_sentence_df.
 ↪text.str.cat()))
page_pred = count_NB_clf.predict(page_count_vector)
page_pred
```

```
array([1])
```

Exercise 4

High similarity would indicate that more components of the vectors are similar to each other and therefore the vectors have a smaller angle between them. For cosines, we know that smaller angles lead to a higher cosine value, leading to a higher score. So a high score indicates high similarity. Students can also be asked to demonstrate this on a mathematical level, but this is optional.

Exercise 5

The process for generating the similarity matrix uses the same code given in the textbook examples, at least for dataframe generation.

```
import tensorflow as tf
import tensorflow_hub as hub

# change this to your own embedding directory
embedding_dir = ""

# load the embedding
embed = hub.load(embedding_dir + "universal-sentence-encoder_4")

text_df["embedding"] = list(embed(text_df.text))

def get_sentence_df(text_df):
    text_df_sentence = []
    text_df_sdg = []
    text_df_text_id = []
    embedding = []
    for (text, sdg, text_id) in iter(zip(text_df.text, text_df.sdg, text_df.text_id)):
        temp_sentence = sent_tokenize(text)
        text_df_sentence = text_df_sentence + temp_sentence
        text_df_sdg = text_df_sdg + [sdg]*len(temp_sentence)
        text_df_text_id = text_df_text_id + [text_id]*len(temp_sentence)
        embedding = embedding + list(embed(temp_sentence))
    sentence_df = pd.DataFrame({"text": text_df_sentence,
                                "sdg": text_df_sdg,
                                "text_id": text_df_text_id,
                                "embedding": embedding})
    return sentence_df

sentence_df = get_sentence_df(text_df)
```

```
2024-07-15 19:47:39.705699: I tensorflow/core/common_runtime/executor.cc:1197] [/
 ↪device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an␣
 ↪error and you can ignore this message): INVALID_ARGUMENT: You must feed a value␣
```

(continues on next page)

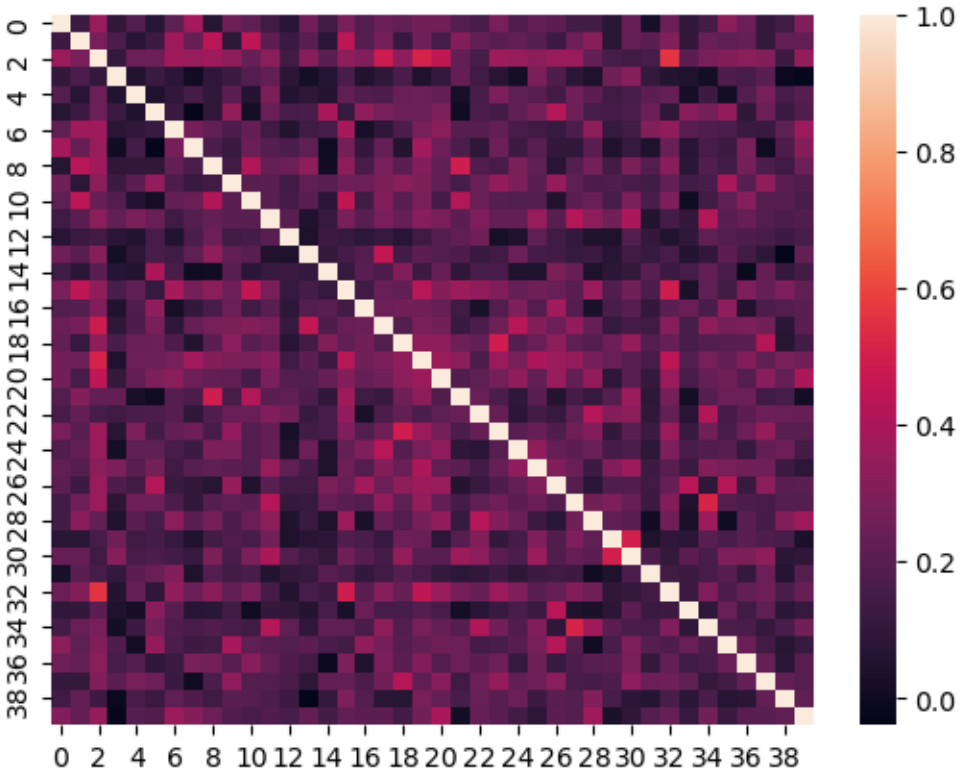
(continued from previous page)

```
↪for placeholder tensor 'inputs' with dtype string
        [[{{node inputs}}]]
```

To get the first 40 documents, we'll use similar code but calling a different dataframe:

```
sns.heatmap(np.array(np.inner(text_df.embedding[:40].tolist(),
                              text_df.embedding[:40].tolist())))
```

```
<Axes: >
```



Exercise 6

The code for this is given below. Students may have trouble with runtimes for this due to the number of sentences, in which case you can subset by SDG or position.

```
sns.heatmap(np.array(np.inner(sentence_df.embedding.tolist(), \
                              sentence_df.embedding.tolist())))
```

Exercise 7

```
def get_most_similar(text_df, sentence, n = 5):
    sentence_sim = np.inner(list(text_df.embedding), embed([sentence]))
    val = sorted(sentence_sim, reverse=True)[n]
    return text_df[sentence_sim > val].text

s = "countries are working hard to save the ocean aminals."
get_most_similar(text_df, sentence = s, n=5)
```

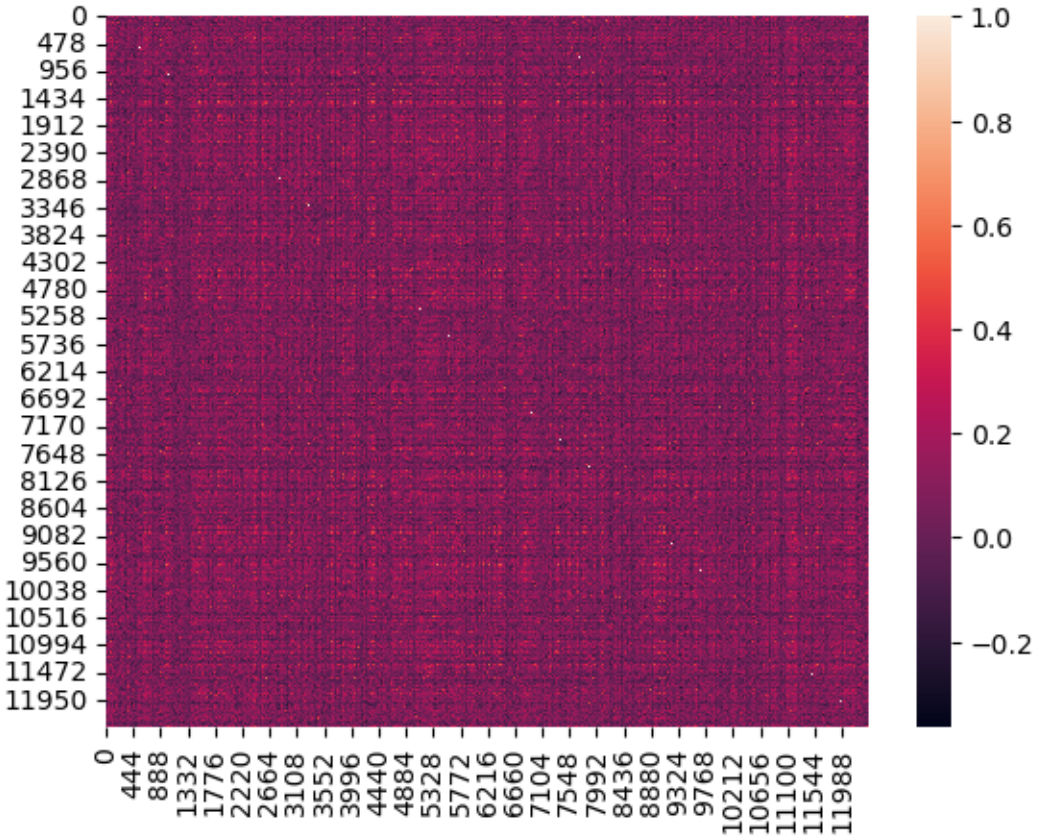
```
5780     Oceans cover approximately 71% of the Earth's ...
8186     It puts forward a working definition of the oc...
12106    However, many commercial fish stocks are overf...
15024    Ninety-eight per cent of the area occupied by ...
18113    The top three African countries that have the ...
Name: text, dtype: object
```

Exercise 8

Students can also take subsets of the sentences, due to the sheer number of sentences present.

```
#get subset of sentence_df with sdg as 1 or 3
sentence_df_subset = sentence_df[sentence_df.sdg.isin([1,3])]

#make a heatmap
sns.heatmap(np.array(np.inner(sentence_df_subset.embedding.tolist(), \
                              sentence_df_subset.embedding.tolist())))
```

```
<Axes: >
```



Exercise 9

```
"""GetMostSimilarToURL(url, text_df, n = 5) takes in a url as a string, parses
it into a single string, and then calls get_most_similar to get the most similar␣
↪documents."""
def GetMostSimilarToURL(url, text_df, n = 5):
    response = requests.get(url)
```

(continues on next page)

(continued from previous page)

```
    soup = BeautifulSoup(response.content, "html.parser")
    page_sentences = []
    for tag in soup.find_all('p'):
        if (len(tag.text)>=30):
            sentences = sent_tokenize(tag.text)
            page_sentences = page_sentences + sentences

    #make all the strings in page_sentences into one string
    page_string = ' '.join(page_sentences)

    #use this to call get_most_similar
    return get_most_similar(text_df, page_string, n)
```

Exercise 10

```
GetMostSimilarToURL("http://gianttortoise.org/en/beyond-tracking", text_df, n = 5)
```

```
4329     A rather large number of species of birds, inv...
4981     In general, neither mechanical nor chemical re...
5758     Assessments now typically involve an “intense ...
10274    The Conservation Banking scheme in the United ...
14146    Local communities and individual farmers prefe...
Name: text, dtype: object
```

```
GetMostSimilarToURL("https://www.dhs.gov/blue-campaign/what-human-trafficking", text_
↪df, n = 5)
```

```
6677     ABSTRACTThis interpretive policy analysis of A...
10997    The trafficking of persons around the world is...
11409    Human trafficking is a serious global problem ...
22162    Sex trafficking is recognized as a national pr...
24426    International human rights law protects agains...
Name: text, dtype: object
```

```
GetMostSimilarToURL("https://www.dol.gov/agencies/odep/program-areas/individuals/
↪older-workers", text_df, n = 5)
```

```
1868     In addition, the government has taken some mea...
2098     Although educational attainment has increased ...
18319    More than 280,000 disabled South Africans aged...
20533    Those who have been brought up in poor homes, ...
24499    Such a shift in viewpoint may account for a la...
Name: text, dtype: object
```

```
GetMostSimilarToURL("https://michigantoday.umich.edu/2022/08/26/positively-breaking-
↪the-age-code/", text_df, n = 5)
```

```
6072     Purpose - The purpose of this paper is to prov...
9517     Eggleston and Fuchs (2012) show that at the be...
13177    The per-capita expenditure is projected accord...
15112    On the other hand, heavy drinking was shown to...
21460    The RAND Future Elderly Model [9] came the clo...
Name: text, dtype: object
```

Answers may vary as to whether or not the returned documents are most similar. However, students should try and call code such as `text_df[6072]` so as to read the texts as much as possible before coming to a conclusion.

14.10 JNB Lab: United States Grant Data

The United States publishes government grant opportunities to solicit eligible opportunities. The dataset of grant opportunities is updated every day and can be found at https://www.grants.gov/xml-extract to be downloaded as an xml file. In this lab, we will be classifying these grant entries into the various UN SDG goals that we talked about throughout the chapter.

14.10.1 Lab Exercises, Part 1: Supervised Learning and Vectorizations

1. Use `pandas` to read in the provided xml file, which is from June 25, 2024, when this lab was first being started. You can use xml files from other days, but the solutions and exercises for this lab are based off of this file.

Now that we've read in the data, we can take a look at it.

```
df.head()
```

```
   OpportunityID                                    OpportunityTitle  \
0         262148  Establishment of the Edmund S. Muskie Graduate...
1         262149  Eradication of Yellow Crazy Ants on Johnston A...
2         131073  Cooperative Ecosystem Studies Unit, Piedmont S...
3         131094  Plant Feedstock Genomics for Bioenergy:  A Joi...
4         131095  Management of HIV-Related Lung Disease and Car...

              OpportunityNumber OpportunityCategory FundingInstrumentType  \
0  SCAPPD-14-AW-161-SCA-08152014                  D                    CA
1                    F14AS00402                  D                    CA
2                    G12AS20003                  D                    CA
3                 DE-FOA-0000598                 D                     G
4                  RFA-HL-12-034                 D                     G

  CategoryOfFundingActivity CategoryExplanation  CFDANumbers  \
0                         O    Public Diplomacy       19.040
1                        NR                None       15.608
2                        ST                None       15.808
3                        ST                None       81.049
4                        HL                None       93.838

   EligibleApplicants                AdditionalInformationOnEligibility  ...  \
0                25.0  Eligibility for U.S. institutions is limited t...  ...
1                99.0  The recipient has already been selected for th...  ...
2                25.0  This financial assistance opportunity is being...  ...
3                99.0  DOE Eligibility Criteria: Applicants from U.S....  ...
4                25.0  Other Eligible Applicants include the followin...  ...

  CloseDateExplanation OpportunityCategoryExplanation  \
0                 None                           None
1                 None                           None
2                 None                           None
3                 None                           None
4                 None                           None
```

(continues on next page)

(continued from previous page)

```
   EstimatedSynopsisPostDate  FiscalYear  EstimatedSynopsisCloseDate  \
0                        NaN         NaN                         NaN
1                        NaN         NaN                         NaN
2                        NaN         NaN                         NaN
3                        NaN         NaN                         NaN
4                        NaN         NaN                         NaN

   EstimatedSynopsisCloseDateExplanation  EstimatedAwardDate  \
0                                   None                 NaN
1                                   None                 NaN
2                                   None                 NaN
3                                   None                 NaN
4                                   None                 NaN

   EstimatedProjectStartDate  GrantorContactName GrantorContactPhoneNumber
0                        NaN                None                       None
1                        NaN                None                       None
2                        NaN                None                       None
3                        NaN                None                       None
4                        NaN                None                       None

[5 rows x 38 columns]
```

2. What variable seems to be the best for which to classify these grant applications into various UN SDGs?
3. Take the variable you've chosen above and transform it into a bag of words matrix.
4. From the bag of words that you've made, identify the first ten features and print them out.
5. Refer to the *section on Document-Term Matrices* to create another vectorization of the documents. Show features 100-110.
6. (Bonus, if you have a strong computer) Create yet another vectorization, similar to the above, but using bigrams. Again, show features 100-110.
7. Once again, make another vectorization, using the TF-IDF Vectorizer. Show features 100-110.
8. Using the function you made in Exercise 4 from Section 5, modify it so as to train a model on the entire UNSDG dataset. Then, use this model to assign predicted classes to each of the entries in the dataset using the Perceptron, Naive Bayes, and Ridge Classifier models.

One thing you've probably noticed is that this dataset does not explicitly have UNSDG classes associated with it. While we can do what we did in the previous exercise for this lab, it is often not the case that we have pre-labeled data. As such, one thing we can turn to is *unsupervised learning*. This was mentioned in the *section giving an overview on machine learning* and involves data that is not already labeled with the "correct" category.

14.10.2 Unsupervised Learning

Two popular algorithms for unsupervised learning revolve around what is known as *clustering*. Clustering, similar to how the name sounds, puts data points into clusters such that there is high similarity within the cluster and low similarity between different clusters. The algorithms for this are a deterministic method called k-means, which puts each data point into a definitive cluster, and a probabilistic method called Gaussian Mixture Modeling, which assigns probabilities for each data point belonging in any cluster.

k-means is described in more detail in the *chapter on linear algebra and optimization*. In practice, however, k-means clustering takes the following procedure:

1. Create k random points to serve as centroids; you choose the value of k. These will be the "centers" of each of our clusters.
2. Assign each existing data point to its closest centroid. This is typically done with Euclidean distance, or

$$\text{dist} = \sqrt{(x_1 - a_1)^2 + (x_2 - a_2)^2 + ... + (x_j - a_j)^2}$$

for data point x and centroid a, each with j features (columns in the dataset).

3. Measure the distance of each point from its assigned centroid and sum all these distances for all n points. Again, this is done with Euclidean distance.
4. Re-calculate the centroid of each cluster by taking the mean vector of all points in the cluster.
5. Repeat steps 2-4 until the total distance metric changes marginally between iterations, or until the centroids do not change position between iterations.

Gaussian Mixture Modeling is similar to k-means, except it assigns probabilities for each point belonging to each cluster, assuming that each point follows a multivariate normal distribution from each cluster's mean point. In practice, it is performed with the Expectation-Maximization (EM) algorithm. Its procedure works as follows:

1. Create k random points to serve as means, and assign each cluster a $j \times j$ covariance matrix; this can be randomly set, but it is more common to use the identity matrix. These mean vectors and covariance matrices are the *mixture parameters*. We also need a prior probability to help with normalization in the next step, which is a single vector of length k detailing the prior probability of a point belonging in any one cluster. This vector is known as the *mixing proportions*.
2. *Expectation Step*: Calculate the log-likelihood of the current data points given these randomly set parameters. This involves two main steps:

- (a) Calculate the probability of each point belonging in each cluster using the multivariate normal probability density function. Following this, multiply these values by the respective probability found in the mixing proportions, then normalize these probabilities so that they sum to 1. These probabilities are stored in a matrix known as a *hidden matrix*.
- (b) Take the cluster of highest probability for each point, then take the natural log of that probability. Perform that over all points, taking the sum of the natural logs.

3. *Maximization Step*: Given the points assigned to each cluster of maximum likelihood, re-calculate the mean vector and covariance matrix for each cluster, as well as the mixing proportions.

- To recalculate mixing proportions, take the cluster where each point has greatest probability to be in and "assign" the point that cluster. Then for each cluster, the new proportion is simply the number of points assigned to that cluster, divided by the total number of points.
- To recalculate mean vectors, for each entry in each mean vector, take the corresponding feature in the data and take the dot product of it to each of the columns in the hidden matrix. Normalize the result based on the sums of the cluster, which gives us a single value for each k; we then do that j times to get the full matrix.
- To recalculate covariance matrices, for each cluster, take the deviation of each point from the mean of the cluster and use these deviations to calculate a new covariance matrix. Then normalize the entries in this covariance matrix by the sums of the probabilities for that cluster.

4. Repeat the Expectation and Maximization Steps until the increase in log-likelihood falls below a specified threshold. Note that various mathematical theories prove that the log-likelihood always increases with each iteration, but at some point this increase will be minimal.

Clusters can be evaluated by a variety of metrics, including consulting domain experts or using Jaccard index. Evaluation of these clusters is beyond the scope of this lab.

14.10.3 Lab Exercises, Part 2: Unsupervised Learning

9. Look at the documentation for k-means and the EM algorithm on `scikit-learn` and use these with various k to cluster the grants. If you have a powerful computer, try $k = 17$ to match the number of UN SDGs.
10. Look at the entries for one of the k-means clusters you made in the previous exercise. How similar are the entries to each other?
11. Compare the entries from the previous cluster you analyzed to a different cluster. How similar are the entries in the first cluster to the ones in the new cluster?
12. (Bonus) Implement k-means and the EM algorithm only using the `numpy` package - feel free to consult other resources for mathematical help. This exercise is not for the faint of heart but is advisable for those who want to improve their understanding of the mathematics underlying these methods!

14.10.4 Lab Exercises, Part 3: Similarity

13. Construct a heatmap for 40 of the grants in the dataset.
14. Isolate twenty of the entries, then write a function that will take one of these twenty entries and an integer k as an input; the function will return the k most similar entries in the rest of the dataset.

14.10.5 Generative AI and Language Models

With the rise of generative AI and large language models like the GPT-system of models developed by OpenAI, it is easier than ever to give a model a string of text and have it classify these texts into predicted UN SDGs.

So how exactly do these models work? The exact mathematical theory behind these models is highly complex as they build on years of research on AI, natural language processing, and machine learning. We mention generative text models briefly in *a previous section*, and state that these models are similar to probabilistic language models but are *generative* in the sense that they will generate the next word in the sequence based on a highly complex model.

We won't have you recreate any generative AI programs here. Instead, we will provide a quick guide through best practices to use them in the context of text classification.

The guiding principle is to **be as specific as possible and narrow the desired task as much as you can.** Expect that you will sometimes get incorrect results, or ones that do not align with the task you intended. As you go, iterate and fine-tune the prompts so that they become more specific.

Additionally, some advanced prompting techniques exist, including few-shot prompting and chain-of-thought prompting, which provides some examples for the LLM or guides the LLM through a few reasoning steps, respectively.

14.10.6 Lab Exercises, Part 4: Generative AI and Language Models

15. Use a LLM available online, such as ChatGPT, and ask it what UNSDGs it predicts some of the grants to fall under using simple question asking. For example, "For a grant with the title _____, what UNSDG aligns best with the grant?" Do the classifications make sense?
16. Use few-shot prompting to classify some other grants. Utilize your own previous classification models (or your own classifications) to provide shots to the prompt. For example, "The grant titled _____ falls under UN SDG _____. The grant titled _____ falls under which UNSDG?" Compare the classifications from this step with the classifications from the previous step.
17. (Bonus) Check out the following additional resources for more details on LLM prompting. Code along with the code provided on the pages and provide your resulting notebooks to answer this question. Extend the principles found in resources to some of these grant titles as well.

- https://cookbook.openai.com/examples/multiclass_classification_for_transactions: a resource from OpenAI, utilizing the capabilities of some of their own models, to classify text documents
- https://huggingface.co/docs/transformers/main/tasks/prompting: a resource from HuggingFace, a package we used in this section, that talks more about general LLM prompting.

14.11 JNB Lab Solutions

The United States publishes government grant opportunities to solicit eligible opportunities. The dataset of grant opportunities is updated every day and can be found at https://www.grants.gov/xml-extract to be downloaded as an xml file. In this lab, we will be classifying these grant entries into the various UN SDG goals that we talked about throughout the chapter.

14.11.1 Lab Exercises, Part 1: Supervised Learning and Vectorizations

1. Use `pandas` to read in the provided xml file, which is from June 25, 2024, when this lab was first being started. You can use xml files from other days, but the solutions and exercises for this lab are based off of this file.

```
import pandas as pd
import numpy as np

#reading the xml file
df = pd.read_xml("GrantsDBExtract20240625v2.xml")
```

Now that we've read in the data, we can take a look at it.

```
df.head()
```

```
   OpportunityID                                  OpportunityTitle  \
0         262148  Establishment of the Edmund S. Muskie Graduate...
1         262149  Eradication of Yellow Crazy Ants on Johnston A...
2         131073  Cooperative Ecosystem Studies Unit, Piedmont S...
3         131094  Plant Feedstock Genomics for Bioenergy:  A Joi...
4         131095  Management of HIV-Related Lung Disease and Car...

              OpportunityNumber OpportunityCategory FundingInstrumentType  \
0  SCAPPD-14-AW-161-SCA-08152014                  D                     CA
1                    F14AS00402                  D                     CA
2                    G12AS20003                  D                     CA
3                 DE-FOA-0000598                 D                      G
4                  RFA-HL-12-034                 D                      G

  CategoryOfFundingActivity CategoryExplanation  CFDANumbers  \
0                         O    Public Diplomacy       19.040
1                        NR                None       15.608
2                        ST                None       15.808
3                        ST                None       81.049
4                        HL                None       93.838

   EligibleApplicants                AdditionalInformationOnEligibility  ...  \
0                25.0  Eligibility for U.S. institutions is limited t...  ...
1                99.0  The recipient has already been selected for th...  ...
2                25.0  This financial assistance opportunity is being...  ...
3                99.0  DOE Eligibility Criteria: Applicants from U.S....  ...
```

(continues on next page)

(continued from previous page)

```
4                25.0  Other Eligible Applicants include the followin...  ...

  CloseDateExplanation OpportunityCategoryExplanation  \
0                 None                          None
1                 None                          None
2                 None                          None
3                 None                          None
4                 None                          None

   EstimatedSynopsisPostDate  FiscalYear  EstimatedSynopsisCloseDate  \
0                        NaN         NaN                         NaN
1                        NaN         NaN                         NaN
2                        NaN         NaN                         NaN
3                        NaN         NaN                         NaN
4                        NaN         NaN                         NaN

   EstimatedSynopsisCloseDateExplanation  EstimatedAwardDate  \
0                                   None                 NaN
1                                   None                 NaN
2                                   None                 NaN
3                                   None                 NaN
4                                   None                 NaN

   EstimatedProjectStartDate  GrantorContactName GrantorContactPhoneNumber
0                        NaN                None                      None
1                        NaN                None                      None
2                        NaN                None                      None
3                        NaN                None                      None
4                        NaN                None                      None

[5 rows x 38 columns]
```

2. What variable seems to be the best for which to classify these grant applications into various UN SDGs?
3. Take the variable you've chosen above and transform it into a bag of words matrix.

```
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.neural_network import MLPClassifier

from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

docs = df.OpportunityTitle
cv = CountVectorizer()
cv_fit = cv.fit_transform(docs)

cv_fit
```

```
<78106x24535 sparse matrix of type '<class 'numpy.int64'>'
        with 762851 stored elements in Compressed Sparse Row format>
```

4. From the bag of words that you've made, identify the first ten features and print them out.

```
feature_names = cv.get_feature_names_out()
feature_names[:10]
```

```
array(['000', '000001', '000001azerbaijan', '000002', '000003', '000008',
       '00001', '000011', '00001395', '00002413'], dtype=object)
```

5. Refer to the *section on Document-Term Matrices* to create another vectorization of the documents. Show features 100-110.

```
count_vectorizer = CountVectorizer(stop_words='english')
count_vector = count_vectorizer.fit_transform(docs).toarray()
count_vector_df_unigram = pd.DataFrame(count_vector, columns=count_vectorizer.get_
↪feature_names_out())
term_freq = pd.DataFrame({"term": count_vector_df_unigram.columns.values, "freq" :␣
↪count_vector_df_unigram.sum(axis=0)})
count_vector_df_unigram.loc[100:110,term_freq.sort_values(by="freq",␣
↪ascending=False)[:10].term] # take a portion
```

	program	research	clinical	national	fy	trial	health	grants	\
100	1	1	0	0	0	0	0	0	
101	0	0	0	0	0	0	0	1	
102	0	0	0	0	0	0	0	0	
103	0	0	0	0	0	0	3	0	
104	0	0	0	0	0	0	0	0	
105	0	0	0	2	0	0	0	0	
106	0	0	0	0	0	0	0	0	
107	0	0	0	0	0	0	0	0	
108	1	0	0	0	0	0	0	0	
109	0	0	0	0	0	0	0	0	
110	0	0	0	1	0	0	0	0	

	development	education
100	0	0
101	1	0
102	0	0
103	0	0
104	0	0
105	1	0
106	0	0
107	0	0
108	0	0
109	0	0
110	0	0

6. (Bonus, if you have a strong computer) Create yet another vectorization, similar to the above, but using bigrams. Again, show features 100-110.

```
#DO NOT RUN THIS CELL UNLESS YOU HAVE BEEFY COMPUTER

count_vectorizer = CountVectorizer(ngram_range=(2,2), stop_words='english')
count_vector = count_vectorizer.fit_transform(docs).toarray()

count_vector_df_bigram = pd.DataFrame(count_vector, columns=count_vectorizer.get_
↪feature_names_out())
term_freq = pd.DataFrame({"term": count_vector_df_bigram.columns.values, "freq" :␣
↪count_vector_df_bigram.sum(axis=0)})
```

(continues on next page)

(continued from previous page)

```
count_vector_df_bigram.loc[100:110,term_freq.sort_values(by="freq",␣
↪ascending=False)[:10].term] # take a portion
```

7. Once again, make another vectorization, using the TF-IDF Vectorizer. Show features 100-110.

```
tfidf_vectorizer = TfidfVectorizer(ngram_range=(1,1), stop_words='english')
tfidf_vector = tfidf_vectorizer.fit_transform(docs).toarray()
tfidf_vector_df = pd.DataFrame(tfidf_vector, columns=tfidf_vectorizer.get_feature_
↪names_out())
term_freq = pd.DataFrame({"term": tfidf_vector_df.columns.values, "freq" : tfidf_
↪vector_df.sum(axis=0)})
tfidf_vector_df.loc[110:120,term_freq.sort_values(by="freq", ascending=False)[:10].
↪term] # take a portion
```

	program	research	national	clinical	grants	fy	health	\
110	0.000000	0.000000	0.273389	0.0	0.000000	0.0	0.0	
111	0.000000	0.000000	0.000000	0.0	0.000000	0.0	0.0	
112	0.167061	0.000000	0.000000	0.0	0.000000	0.0	0.0	
113	0.200372	0.000000	0.000000	0.0	0.000000	0.0	0.0	
114	0.000000	0.000000	0.000000	0.0	0.000000	0.0	0.0	
115	0.000000	0.000000	0.000000	0.0	0.000000	0.0	0.0	
116	0.000000	0.133704	0.000000	0.0	0.000000	0.0	0.0	
117	0.000000	0.000000	0.000000	0.0	0.000000	0.0	0.0	
118	0.000000	0.000000	0.000000	0.0	0.000000	0.0	0.0	
119	0.000000	0.000000	0.000000	0.0	0.304318	0.0	0.0	
120	0.000000	0.000000	0.000000	0.0	0.000000	0.0	0.0	

	development	cooperative	trial
110	0.000000	0.0	0.0
111	0.000000	0.0	0.0
112	0.000000	0.0	0.0
113	0.000000	0.0	0.0
114	0.161333	0.0	0.0
115	0.000000	0.0	0.0
116	0.000000	0.0	0.0
117	0.148290	0.0	0.0
118	0.000000	0.0	0.0
119	0.000000	0.0	0.0
120	0.000000	0.0	0.0

8. Using the function you made in Exercise 4 from Section 5, modify it so as to train a model on 90% of the UNSDG dataset. Then, use this model to assign predicted classes to each of the entries in the dataset using the Perceptron, Naive Bayes, and Ridge Classifier models.

This solution only uses Naive Bayes for simplicity; the other algorithms can easily replace the `nb` *variable in this solution.*

```
# change this to your own data directory
data_dir = "data/"

# read and preprocess data
text_file_name = "osdg-community-data-v2023-01-01.csv"
text_df = pd.read_csv(data_dir + text_file_name,sep = "\t",  quotechar='"')
col_names = text_df.columns.values[0].split('\t')
text_df[col_names] = text_df[text_df.columns.values[0]].apply(lambda x: pd.
↪Series(str(x).split("\t")))
text_df = text_df.astype({'sdg':int, 'labels_negative': int, 'labels_positive':int,
```

(continues on next page)

(continued from previous page)

```
 ↪'agreement': float}, copy=True)
text_df.drop(text_df.columns.values[0], axis=1, inplace=True)
text_df = text_df.query("agreement > 0.5 and (labels_positive - labels_negative) > 2")
text_df.reset_index(inplace=True, drop=True)

docs = text_df.text
categories = text_df.sdg
X_train, X_test, y_train, y_test = \
    train_test_split(docs, categories, test_size=0.1, random_state=7)

X_train_count_vectorizer = CountVectorizer(ngram_range=(2,2), stop_words = "english" )
X_train_count_vectorizer.fit(X_train)
X_train_count_vector = X_train_count_vectorizer.transform(X_train)

#train Naive Bayes model
nb = MultinomialNB()
nb.fit(X_train_count_vector, y_train)

#apply model to the grant titles
X_test_count_vector = X_train_count_vectorizer.transform(docs)
y_pred = nb.predict(X_test_count_vector)
print(y_pred)
```

```
[ 5  3 10 ...  4  6  3]
```

One thing you've probably noticed is that this dataset does not explicitly have UNSDG classes associated with it. While we can do what we did in the previous exercise for this lab, it is often not the case that we have pre-labeled data. As such, one thing we can turn to is *unsupervised learning*. This was mentioned in the *section giving an overview on machine learning* and involves data that is not already labeled with the "correct" category.

14.11.2 Unsupervised Learning

Two popular algorithms for unsupervised learning revolve around what is known as *clustering*. Clustering, similar to how the name sounds, puts data points into clusters such that there is high similarity within the cluster and low similarity between different clusters. The algorithms for this are a deterministic method called k-means, which puts each data point into a definitive cluster, and a probabilistic method called Gaussian Mixture Modeling, which assigns probabilities for each data point belonging in any cluster.

k-means is described in more detail in the *chapter on linear algebra and optimization*. In practice, however, k-means clustering takes the following procedure:

1. Create k random points to serve as centroids; you choose the value of k. These will be the "centers" of each of our clusters.
2. Assign each existing data point to its closest centroid. This is typically done with Euclidean distance, or

$$\text{dist} = \sqrt{(x_1 - a_1)^2 + (x_2 - a_2)^2 + ... + (x_j - a_j)^2}$$

for data point x and centroid a, each with j features (columns in the dataset).

3. Measure the distance of each point from its assigned centroid and sum all these distances for all n points. Again, this is done with Euclidean distance.
4. Re-calculate the centroid of each cluster by taking the mean vector of all points in the cluster.
5. Repeat steps 2-4 until the total distance metric changes marginally between iterations, or until the centroids do not change position between iterations.

Gaussian Mixture Modeling is similar to k-means, except it assigns probabilities for each point belonging to each cluster, assuming that each point follows a multivariate normal distribution from each cluster's mean point. In practice, it is performed with the Expectation-Maximization (EM) algorithm. Its procedure works as follows:

1. Create k random points to serve as means, and assign each cluster a $j \times j$ covariance matrix; this can be randomly set, but it is more common to use the identity matrix. These mean vectors and covariance matrices are the *mixture parameters*. We also need a prior probability to help with normalization in the next step, which is a single vector of length k detailing the prior probability of a point belonging in any one cluster. This vector is known as the *mixing proportions*.
2. *Expectation Step*: Calculate the log-likelihood of the current data points given these randomly set parameters. This involves two main steps:

- (a) Calculate the probability of each point belonging in each cluster using the multivariate normal probability density function. Following this, multiply these values by the respective probability found in the mixing proportions, then normalize these probabilities so that they sum to 1. These probabilities are stored in a matrix known as a *hidden matrix*.
- (b) Take the cluster of highest probability for each point, then take the natural log of that probability. Perform that over all points, taking the sum of the natural logs.

3. *Maximization Step*: Given the points assigned to each cluster of maximum likelihood, re-calculate the mean vector and covariance matrix for each cluster, as well as the mixing proportions.

- To recalculate mixing proportions, take the cluster where each point has greatest probability to be in and "assign" the point that cluster. Then for each cluster, the new proportion is simply the number of points assigned to that cluster, divided by the total number of points.
- To recalculate mean vectors, for each entry in each mean vector, take the corresponding feature in the data and take the dot product of it to each of the columns in the hidden matrix. Normalize the result based on the sums of the cluster, which gives us a single value for each k; we then do that j times to get the full matrix.
- To recalculate covariance matrices, for each cluster, take the deviation of each point from the mean of the cluster and use these deviations to calculate a new covariance matrix. Then normalize the entries in this covariance matrix by the sums of the probabilities for that cluster.

Clusters can be evaluated by a variety of metrics, including consulting domain experts or using Jaccard index. Evaluation of these clusters is beyond the scope of this lab.

14.11.3 Lab Exercises, Part 2: Unsupervised Learning

9. Look at the documentation for k-means and the EM algorithm on `scikit-learn` and use these with various k to cluster the grants. If you have a powerful computer, try $k = 17$ to match the number of UN SDGs.

```
#answers for this will likely take the form of the following:
#for k-means:
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaled_data = scaler.fit_transform(count_vector_df_unigram)
kmeans = KMeans(n_clusters=5, random_state=0).fit_predict(scaled_data)

#for Gaussian Mixture Modeling with EM Algorithm:
from sklearn.mixture import GaussianMixture
gm = GaussianMixture(n_components=5, random_state=0).fit_predict(scaled_data)
```

10. Look at the entries for one of the k-means clusters you made in the previous exercise. How similar are the entries to each other?

Answers will vary due to the random nature of both clustering methods.

11. Compare the entries from the previous cluster you analyzed to a different cluster. How similar are the entries in the first cluster to the ones in the new cluster?

Answers will vary both due to the random nature of clustering, as well as depending on the chosen clusters.

12. (Bonus) Implement k-means and the EM algorithm only using the `numpy` package - feel free to consult other resources for mathematical help. This exercise is not for the faint of heart but is advisable for those who want to improve their understanding of the mathematics underlying these methods!

Implementations may vary. One example implementation is given below.

```
#For k-means:
import copy
def kmeans(data, k):
    #so we can't have k less than 1, or there will be problems.
    if k < 1:
        print("Invalid k-value given; k must be at least 1")
        return -1

    #make a dictionary to hold the classes of all the items.
    cluster_assignments = initial_new_cluster(k)

    #we also need something to hold our cluster evaluations
    evals = []

    #set our initializations
    inits = get_randinits(data, k)

    #main clustering process
    clustered = False
    #a max iterations parameter for sanity check
    maxRuns = 50
    numRuns = 0

    while clustered == False and numRuns < maxRuns:
        prev_clusters = copy.deepcopy(cluster_assignments)
        cluster_assignments = initial_new_cluster(k)
        for entry in data:
            dists = get_distances(entry, inits)
            minDist = np.min(dists)
            closest_cluster = np.where(dists == minDist)[0][0]
            cluster_assignments[closest_cluster].append(entry)

        new_centers = get_new_centers(cluster_assignments, inits)
        #check: are our centers the same?
        #if clusters_same(prev_clusters, cluster_assignments):
        if centroids_same(inits, new_centers):
            clustered = True
        else:
            inits = copy.deepcopy(new_centers)
            evals.append(eval_clusters(cluster_assignments, inits))
        numRuns += 1

    return cluster_assignments, evals

def get_randinits(data, k):
    #set min and max values per variable
```

(continues on next page)

(continued from previous page)

```
    minvals = []
    maxvals = []

    #range over length of first vector
    for i in range(len(data[0])):
        current_min = data[0][i]
        current_max = data[0][i]

        #now range over all rows:
        for j in range(len(data)):
            if data[j][i] < current_min:
                current_min = data[j][i]
            if data[j][i] > current_max:
                current_max = data[j][i]

        minvals.append(current_min)
        maxvals.append(current_max)

    #okay, now we have a list of the min/max values.
    #let's make random vectors with np.random.rand(k, len(minvals))
    #to each one, we'll multiply it by the range (min - max) and add the min.
    inits = np.random.rand(k, len(minvals))
    for i in range(len(minvals)):
        current_range = maxvals[i] - minvals[i]
        inits[:, i] *= current_range
        inits[:, i] += minvals[i]

    return inits

#two helper functions: get distances, and clusters same.
def get_distances(entry, inits):
    dists = []
    for item in inits:
        currentDist = get_dist(entry, item)
        dists.append(currentDist)
    return np.array(dists)

def get_dist(entry, item):
    dist = 0.0
    for i in range(len(item)):
        dist += (entry[i] - item[i]) ** 2
    return np.sqrt(dist)

def centroids_same(c1, c2):
    same = np.all(c1 == c2)
    return same

def get_new_centers(clusters, centers):
    #we'll get the data, and the number of clusters
    newcenters = []
    clusts = clusters.keys()
    for c in clusts:
        current_clust = np.array(clusters[c])
        #for each cluster, we're going to average each variable
        new_center = []
        if len(current_clust) == 0:
            newcenters.append(centers[c])
```

(continues on next page)

(continued from previous page)

```
        else:
            for i in range(len(current_clust[0])):
                new_center.append(np.average(current_clust[:, i]))
            newcenters.append(new_center)

    return np.array(newcenters)

def eval_clusters(clusters, centroids):
    score = 0
    for i in range(len(centroids)):
        current_clust = clusters[i]
        for j in current_clust:
            score += get_dist(j, centroids[i])

    return score

def initial_new_cluster(k):
    cluster_assignments = {}
    for i in range(k):
        cluster_assignments[i] = []
    return cluster_assignments
```

```
#For GMM:
def MultiVarNormal(x, mean, cov):
    """
    MultiVarNormal implements the PDF for a multivariate gaussian distribution
    (one sample at a time)
    Input:
        - x: (d,) numpy array
        - mean: (d,) numpy array; the mean vector
        - cov: (d,d) numpy array; the covariance matrix
    Output:
        prob - a scaler
    Hint:
        - Use np.linalg.det to compute determinant
        - Use np.linalg.pinv to invert a matrix
    """
    d = x.shape[0]
    pdf = 1
    # begin solution
    #added: another thing to keep in mind is that sometimes we get determinants very␣
 ↪close to 0
    #this makes probabilities very large; much larger than 1
    #so we need to compute a pseudo-derivative if it is too small.
    mydet = np.linalg.det(cov)
    #if mydet < 1e-12:
        #pseudo determinant through evals
        #evals, evecs = np.linalg.eig(cov)
        #mydet = np.product(evals[evals > 1e-12])
        #d = np.linalg.matrix_rank(cov)

    n1 = 1.0 / np.sqrt(((2.0 * np.pi) ** d) * mydet)
    n2 = np.exp(-0.5 * (np.matmul(np.transpose(x - mean), np.matmul(np.linalg.
 ↪pinv(cov), (x - mean)))))
    pdf = n1*n2
    # end solution
```

(continues on next page)

(continued from previous page)

```
    return pdf

def UpdateMixProps(hidden_matrix):
    """
    Returns the new mixing proportions given a hidden matrix
    Input:
        - hidden_matrix: (n, k) numpy array
    Output:
        - mix_props: (k,) numpy array
    Hint:
        - See equation in Lecture 10 pg 42
    """
    n,k = hidden_matrix.shape
    mix_props = np.zeros(k)
    # begin solution
    for i in range(k):
        current_mean = np.mean(hidden_matrix[:,i])
        mix_props[i] = current_mean

    # end solution
    return mix_props

def UpdateMeans(X, hidden_matrix):
    """
    Update means for gaussian distributions given the data and the hidden matrix
    Input:
        - X: (n, d) numpy array
        - hidden_matrix: (n, k) numpy array
    Output:
        - new_means: (k, d) numpy array
    Hint:
        - See equation in Lecture 10 pg 43
    """
    n,d = X.shape
    k = hidden_matrix.shape[1]
    new_means = np.zeros([k,d])
    # begin solution
    #so to think about this a little bit because for some reason I am not getting␣
↪this:
    #we need to, for each entry in each mean vector, take the corresponding entry in␣
↪the data (entry = feature)
    #then we need to dot product it by each of the columns in the hidden matrix
    #and then normalize based on the sums of the cluster, which gives us a single␣
↪value for each k
    #then do that d times to get the full matrix.
    for i in range(d): #across the columns
        current_subset = X[:,i]
        for c in range(k): #across the clusters
            current_vals = hidden_matrix[:,c]

            weighted_val = np.dot(current_subset, current_vals)
            total_prob = np.sum(current_vals)

            current_entry = weighted_val / total_prob

            new_means[c,i] = current_entry
    # end solution
```

(continues on next page)

(continued from previous page)

```
    return new_means

def UpdateCovars(X, hidden_matrix, means):
    """
    Update covariance matrices for gaussian distributions given the data and the␣
 ↪hidden matrix
    Input:
        - X: (n, d) numpy array
        - hidden_matrix: (n, k) numpy array
        - means: (k, d) numpy array; means for all distributions
    Output:
        - new_covs: (k, d, d) numpy array
    Hint:
        - See equation in Lecture 10 pg 43
    """
    n,d = X.shape
    k = hidden_matrix.shape[1]
    new_covs = np.zeros([k,d,d])
    # begin solution
    #to think about this:
    #first we want to range through each cluster
    for i in range(k):
        #we want to get the sum of probabilities for this cluster
        current_sum = np.sum(hidden_matrix[:,i])

        #next we want to range through each data point
        for j in range(n):
            #take the deviation from the mean of the cluster
            current_diff = X[j] - means[i]

            #take the probability for this data point from this cluster
            current_prob = hidden_matrix[j, i]

            #need to get the matrix - np.matmul does not work here
            #so we need to brute force this
            for a in range(d):
                for b in range(d):
                    current_cov = current_diff[a] * current_diff[b] * current_prob
                    new_covs[i][a,b] += current_cov

        #then normalize the entries
        new_covs[i] = new_covs[i] / current_sum
    # end solution
    return new_covs

def HiddenMatrix(X, means, covs, mix_props):
    """
    Computes the hidden matrix for the data.
    This function should also compute the log likelihood
    Input:
        - X: (n, d) numpy array
        - means: (k, d) numpy array; the mean vectors
        - new_covs: (k, d, d) numpy array; the covariance matrices
        - mix_props: (k,) numpy array; the mixing proportions
    Output:
        - hidden_matrix: (n, k) numpy array
        - ll: scalar; the log likelihood
```

(continues on next page)

(continued from previous page)

```
    Hint:
        - Construct an intermediate matrix t of shape (n, k).
            t[i,j] = P(X_i | c = j)P(c = j), for i=1,...,n, j=1,...,k
            This matrix can be used to calculate the loglikelihood and the hidden␣
↪matrix.
        - Each row of the hidden matrix sums to 1
        - hidden_matrix[i,j] = P(X_i | c = j)P(c = j) / (Sum_{l=1}^{k}(P(X_i | c =␣
↪l)P(c = l))),
            for i=1,...,n, j=1,...,k
    """
    n,d = X.shape
    k = means.shape[0]
    hidden_matrix = np.zeros([n,k])
    ll = 0
    t = np.zeros([n,k]) # intermediate matrix

    # begin solution
    for i in range(n): #across the rows
        for c in range(k): #across the clusters
            current_mean = means[c]
            current_cov = covs[c]
            current_mix = mix_props[c]

            current_prob = MultiVarNormal(X[i, :], current_mean, current_cov)
            #current_prob = stats.multivariate_normal.pdf(X[i, :], current_mean,␣
↪current_cov)
            current_entry = current_prob * current_mix

            hidden_matrix[i,c] = current_entry

    #we have the intermediate matrix, we now need to normalize so these entries sum␣
↪to 1
    for i in range(n):
        current_total = np.sum(hidden_matrix[i,:])
        for j in range(k): #across the columns, which is the clusters for hidden␣
↪matrix
            current_val = hidden_matrix[i,j] / current_total
            hidden_matrix[i,j] = current_val

    #now to take the log likelihood via the matrix
    for a in range(n):
        #we want the log-likelihood of the biggest value
        b = np.argmax(hidden_matrix[a,:])
        ll += np.log(hidden_matrix[a,b])

    # end solution
    return hidden_matrix,ll

def GMM(X, init_means, init_covs, init_mix_props, thres=0.001):
    """
    Runs the GMM algorithm
    Input:
        - X: (n, d) numpy array
        - init_means: (k,d) numpy array; the initial means
        - init_covs: (k,d,d) numpy arry; the initial covariance matrices
        - init_mix_props: the initial mixing proportions
    Output:
```

(continues on next page)

(continued from previous page)

```
        - clusters: (n,) numpy array; the cluster assignment for each sample
        - hidden_matrix: (n, k) numpy array
        - ll_list: the log likelihood at all iterations
    Hint:
        - Use all above functions
        - Stoping condition: the difference between your ll from the current iteration
            and the last iteration is below your threshold
        - You can set maximum iteration as 1,000 to avoid infinite loop
        - Remember to check if your algorithm has converged
    """
    n,d = X.shape
    k = init_means.shape[0]
    clusters = np.zeros(n)
    ll_list = []
    # begin solution
    threshold_met = False
    current_iter = 0
    current_means = init_means
    current_covs = init_covs
    current_mix = init_mix_props
    while threshold_met != True:
        #first: get the hidden matrix and log-likelihood
        current_hidden, current_ll = HiddenMatrix(X, current_means, current_covs,␣
↪current_mix)
        ll_list.append(current_ll)

        #second: update all parameters
        current_means = UpdateMeans(X, current_hidden)
        current_covs = UpdateCovars(X, current_hidden, current_means)
        current_mix = UpdateMixProps(current_hidden)

        #third: check if our threshold is met
        #or if we have only run one time.
        if current_iter == 0:
            current_iter += 1
        elif np.abs(ll_list[current_iter] - ll_list[current_iter - 1]) < thres:
            threshold_met = True
        else:
            current_iter += 1

    #now we need to put each point into a cluster
    for j in range(n):
        largest_prob_cluster = np.argmax(current_hidden[j, :])
        clusters[j] = largest_prob_cluster
    # end solution
    return clusters,current_hidden,ll_list
```

14.11.4 Lab Exercises, Part 3: Similarity

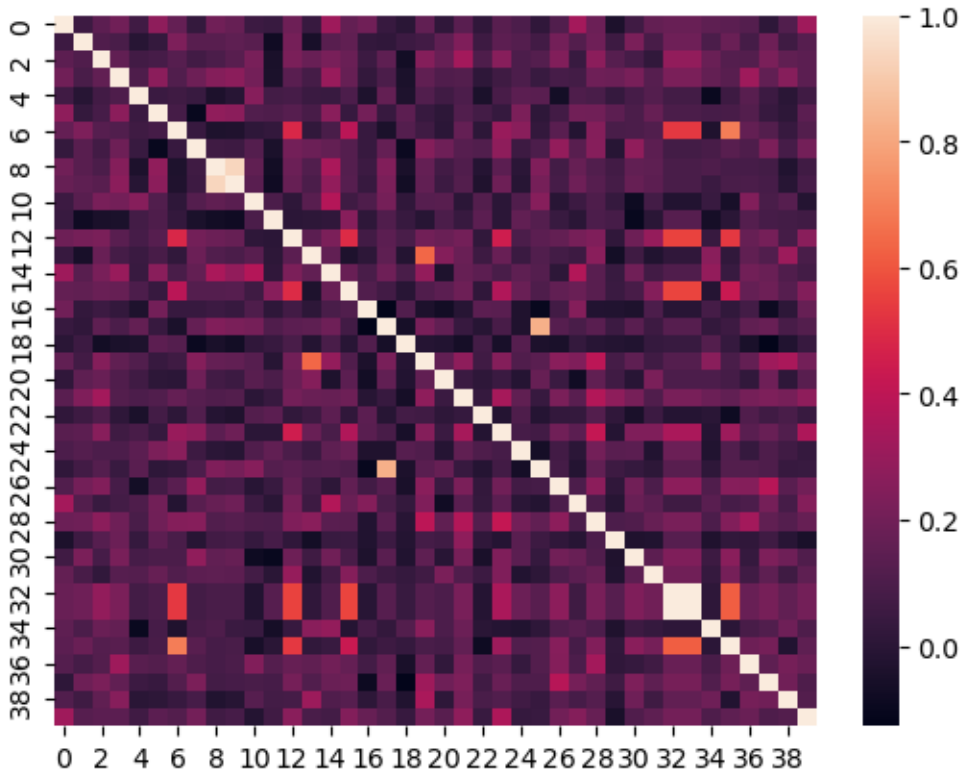
13. Construct a heatmap for 40 of the grants in the dataset.

```
/var/folders/px/b7vc3nh913zb_m0x36ncftj00000gn/T/ipykernel_68678/3899229267.py:6:␣
↪DtypeWarning: Columns (33,36,37) have mixed types. Specify dtype option on␣
↪import or set low_memory=False.
  df = pd.read_csv("GrantsDBExtract20240625v2.csv")[:10000]
```

```
import tensorflow as tf
import tensorflow_hub as hub

# change this to your own embedding directory
embedding_dir = ""

# load the embedding
embed = hub.load(embedding_dir + "universal-sentence-encoder_4")

df["embedding"] = list(embed(df.OpportunityTitle))

#make heatmap of similarities
import seaborn as sns
sns.heatmap(np.array(np.inner(df.embedding[:40].tolist(),
                              df.embedding[:40].tolist())))
```

```
2024-07-15 21:10:45.922395: I tensorflow/core/common_runtime/executor.cc:1197] [/
↪device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an␣
↪error and you can ignore this message): INVALID_ARGUMENT: You must feed a value␣
↪for placeholder tensor 'inputs' with dtype string
         [[{{node inputs}}]]
```

```
<Axes: >
```



14. Isolate twenty of the entries, then write a function that will take one of these twenty entries and an integer k as an input; the function will return the k most similar entries in the rest of the dataset.

The function and its setup are given as follows:

```
test_set = df[:20]
train_set = df[20:]

def get_most_similar(text_df, point, n = 5):
    sentence = point.OpportunityTitle
    sentence_sim = np.inner(list(text_df.embedding), embed([sentence]))
    val = sorted(sentence_sim, reverse=True)[n]
    return text_df[sentence_sim > val].OpportunityTitle

print("Grants most similar to grant", test_set.iloc[0].OpportunityTitle)
get_most_similar(train_set, test_set.iloc[0])
```

```
Grants most similar to grant Establishment of the Edmund S. Muskie Graduate␣
↪Internship Program
```

```
427     Internship Training Program Interior Museum Pr...
3015                  Career Discovery Internship Program
7994                Mosaics in Science Internship Program
7995                Mosaics in Science Internship Program
8964                      Buiness Plan Internship Program
Name: OpportunityTitle, dtype: object
```

14.11.5 Generative AI and Language Models

With the rise of generative AI and large language models like the GPT-system of models developed by OpenAI, it is easier than ever to give a model a string of text and have it classify these texts into predicted UN SDGs.

So how exactly do these models work? The exact mathematical theory behind these models is highly complex as they build on years of research on AI, natural language processing, and machine learning. We mention generative text models briefly in *a previous section*, and state that these models are similar to probabilistic language models but are *generative* in the sense that they will generate the next word in the sequence based on a highly complex model.

We won't have you recreate any generative AI programs here. Instead, we will provide a quick guide through best practices to use them in the context of text classification.

The guiding principle is to **be as specific as possible and narrow the desired task as much as you can.** Expect that you will sometimes get incorrect results, or ones that do not align with the task you intended. As you go, iterate and fine-tune the prompts so that they become more specific.

Additionally, some advanced prompting techniques exist, including few-shot prompting and chain-of-thought prompting, which provides some examples for the LLM or guides the LLM through a few reasoning steps, respectively.

14.11.6 Lab Exercises, Part 4: Generative AI and Language Models

15. Use a LLM available online, such as ChatGPT, and ask it what UNSDGs it predicts some of the grants to fall under using simple question asking. For example, "For a grant with the title _____, what UNSDG aligns best with the grant?" Do the classifications make sense?

Answers may vary but should make sense if an advanced LLM is used.

16. Use few-shot prompting to classify some other grants. Utilize your own previous classification models (or your own classifications) to provide shots to the prompt. For example, "The grant titled _____ falls under UN SDG _____. The grant titled _____ falls under which UNSDG?" Compare the classifications from this step with the classifications from the previous step.

Answers may vary but should make sense if an advanced LLM is used. Prompts should also follow the format specified in this question.

17. (Bonus) Check out the following additional resources for more details on LLM prompting. Code along with the code provided on the pages and provide your resulting notebooks to answer this question. Extend the principles found in resources to some of these grant titles as well.

    - https://cookbook.openai.com/examples/multiclass_classification_for_transactions: a resource from OpenAI, utilizing the capabilities of some of their own models, to classify text documents
    - https://huggingface.co/docs/transformers/main/tasks/prompting: a resource from HuggingFace, a package we used in this section, that talks more about general LLM prompting.

Answers should follow the code examples given on the websites.

CHAPTER 15

COMPLEX SYSTEMS - Wheaton College Team

Note

This chapter on complex systems wsa developed by Dr. Paul Isihara (lead professor) with Dr. Peter Jantsch (assisting professor) and Emma Brown, Taylor Chiles, Jonathan Contreras, Joey Geisz, Luke Hamm, Jakob Hildebrandt, Jadon Hsu, Aidan Kingsbury, Catie Lamberson, Sally Li, Avery McKeaney, Gavin McClowry, Hannah Mullins, Kendra Reimer, Jocelyn Rosa, Cassie Seaver, Harrison Stanton, Tyler Weiner (Wheaton College students).

15.1 Introduction

15.1.1 Setting the Scene

A brutal two year civil war (Nov 2020-Nov 2022) crippled the entire infrastructure of the Tigray region of northern Ethiopia. All 6-7 million people in the region experienced trauma due to a large number of civilian deaths and victims of rape as a weapon of war, as well as a 2-year blockade of internet, cell phone communication, banking, and humanitarian aid. Today, there are food shortages everywhere, and hundreds of deaths by starvation have occurred during the writing of this chapter. Poverty abounded everywhere during the war and continues to be a major problem with many women forced into prostitution for survival. Almost all medical services were incapacitated and doctors were among those left begging on the streets for food. In the education sector, over 2,000 schools were closed during the war. Without outside intervention, over two million children faced their third year out of school. To compound this already overwhelming scenario, one year after the Pretoria peace agreement was signed, fighting continued in the western part of Tigray, leaving 1 million internally displaced people (IDP) living in over 600 IDP camps scattered across the rest of the region (**Figure 1**). Maintaining a minimum holistic standard of living for one million IDP (food, water, sanitation, health, education, safety,…) is a complex problem. Is this a problem for the Tigray regional government to implement a well-orchestrated and funded policy across the region? (With all the other problems it faces, how high a priority can the regional government give to the IDP's?) Or, is this a problem requiring intervention by individuals, NGOs, and local district governments (woredas) who might make a difference for selected groups of IDP's?

We connected with a Chicago-based NGO which visited a small IDP camp in the town of Nebelet about 3 hours drive northwest of the regional capital Mekelle. After learning first-hand about the needs of this

one camp, in order to better understand the overall needs of the IDPs, we considered a dataset published by the U.N.'s International Office for Migration (IOM) in August 2023 (https://data.humdata.org/dataset/ethiopia-displacement-northern-region-tigray-idps-site-assessment-iom-dtm?

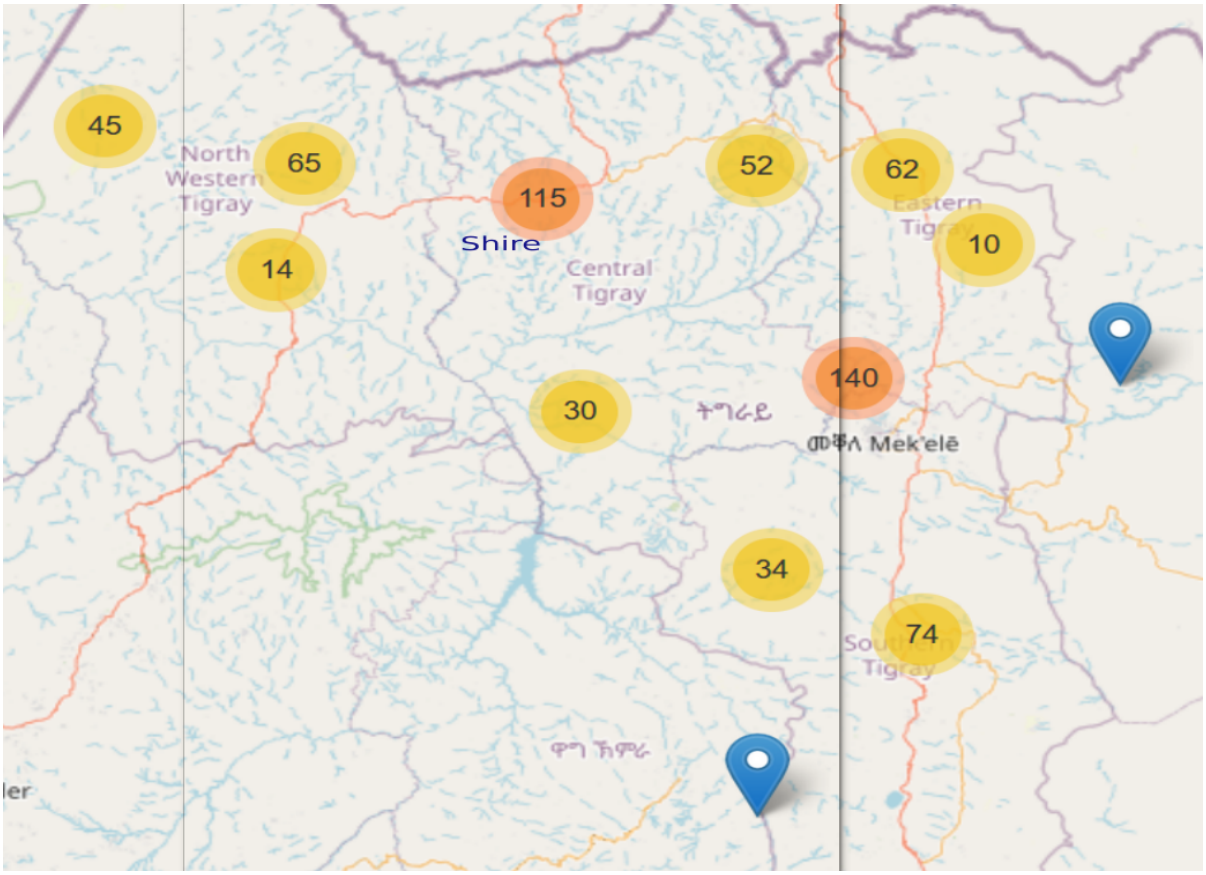


Figure 1 Numbers of IDP camps in Tigray. The regional capital (Mekelle) and second largest city (Shire) have the largest number of camps. The number and needs of the IDP camps vary in the different administrative zones of Tigray.

The needs of the IDPs are complex. A question we pose here and re-visit at the end of the chapter is: "how can we rank the administrative zones according to the need of their IDP camps?"

15.1.2 Complex System Science

This chapter offers a brief introduction to complex system science. Complexity is an intuitive concept which arises across disciplines, including urban issues such as the problem of an influx of large numbers of displaced people. The science of complex systems seeks to develop a rigorous methodology for analyzing overall system behavior based on interactions involving a large number of constituent parts or agents. Analytical methods, including those arising from statistical physics, utilize a wide range of undergraduate level mathematics such as calculus, linear algebra, differential equations, and probability and statistics. Our goal is to show how a few important concepts found in these quantitative methods (for example, equilibrium, phase transition, and entropy) support basic notions of complex systems, and how they offer a perspective on practical questions in complex human society.

From a humanitarian standpoint, two goals of complex system science include:

- prevent dangerous misunderstanding and unjust practices associated with an erroneous view of complex social/environmental systems; and
- provide insights which can help facilitate enduring solutions to complex societal problems.

A classic example of a dangerous misunderstanding of a complex situation in Chicago is the urban redlining maps from the 1920's which classified neighborhoods with high density of African Americans as 'hazardous' and deemed mortgages not insurable by the F.H.A. Redlining permanently crippled Black home ownership (see **Figure 2**, left panel). As for a recent example of a helpful insight into the same urban area, Robert Sampson [2012] introduced a concept called *collective efficacy* (cooperation between community members) suggesting that violent crime reduction might be facilitated by a coherent community-based approach (**Figure 2** right panel).

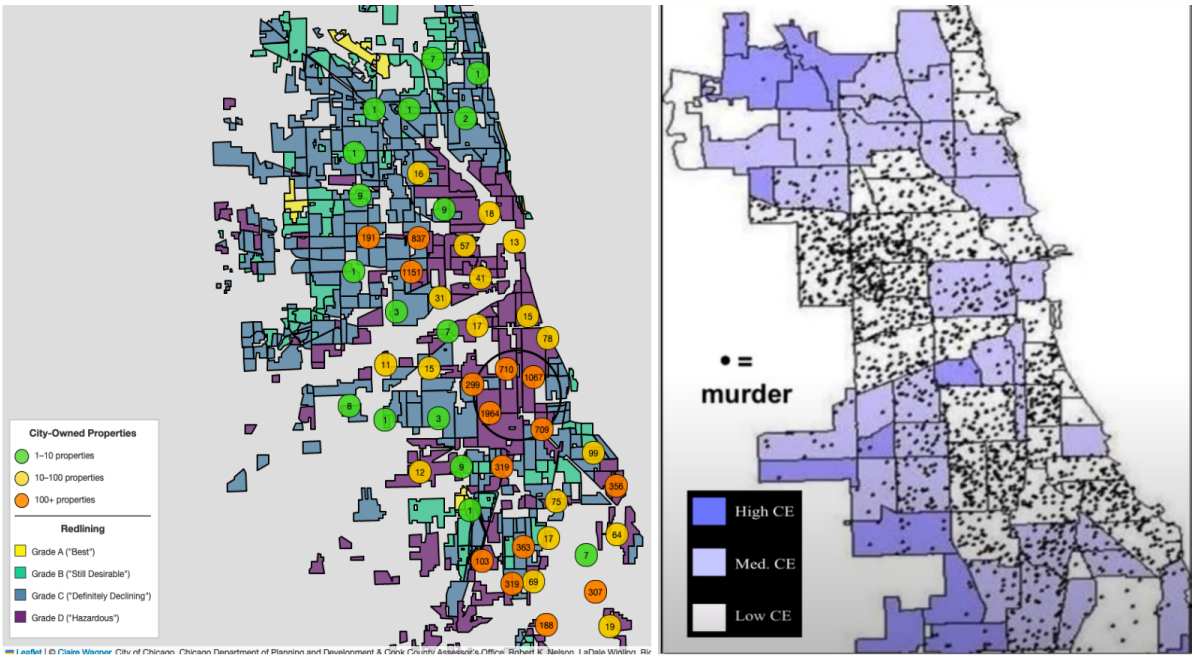


Figure 2 (Left) Home Owners' Loan Corporation neighborhood classifications and present day city owned empty lots. The circled area with a high concentration of empty lots was labeled 'Hazardous'. Map Credit: Claire Wagner (Right) Collective efficacy (CE) levels and homicides. Map credit: "Great American City: Chicago & the Enduring Neighborhood Effect." YouTube, uploaded by Robert Sampson, 18 March 2014, https://www.youtube.com/watch?v=gEXSOUqwztA. Permissions: YouTube Terms of Service

)

This chapter is organized into three sections:

- Fundamental Concepts (**Section 2**);
- Mathematical Formalism from Statistical Physics (**Section 3**); and
- Societal Applications (**Section 4**).

The Exercises are an integral part of this chapter, and should at least be read if not attempted in earnest (solutions to the Exercises are included in **Section 5**). A Python Jupyter Notebook (JNB) lab with solutions appear at the end of the chapter.

15.2 Fundamental Concepts

In this section we describe qualitatively several concepts fundamental to complex systems.

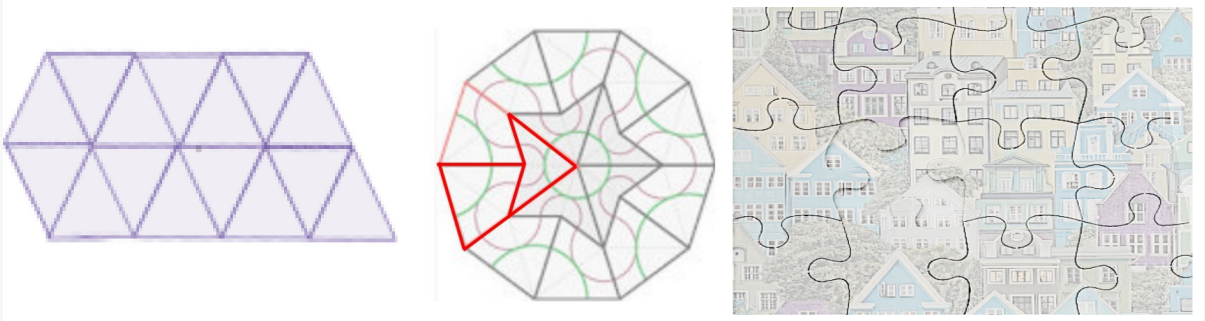
15.2.1 Homogeneity of Constituent Parts

One basic difference between pure physical science and a study of human society is an assumption about the degree of homogeneity of the constituent parts of a system. Statistical physics methodology applied to an ensemble of identical gas molecules is simpler than application to a society comprised of unique individuals with different values, decision-making, propensity to do good or evil, etc. Systems where a homogeneity assumption of the constituent units is reasonable (eg. large flock of birds in synchronized flight) are more amenable to mathematical approaches.

Exercise

Exercise

2.1 Referring to Figure 3, explain how the degree of homogeneity of the constituent jig saw pieces relates to the overall complexity of the jigsaw puzzle.



Left: 1 piece jigsaw puzzle (crystal) Center: 2 piece 'kite and dart' jigsaw puzzle (quasicrystal) Right: Random piece jigsaw puzzle of an urban neighborhood.

15.2.2 System Classification

Open, Closed, and Isolated Systems

Systems can be classified according to whether they exchange energy or mass with the environment:

- ISOLATED SYSTEM: exchanges neither mass nor energy with its environment.
- CLOSED SYSTEM: exchanges energy but not mass with its environment.
- OPEN SYSTEM: exchanges both mass and energy with its environment.

Note that in studying complex human society, energy exchange may be broadly defined (for example, a remote exchange of ideas or financial credit.)

Random, Correlated and Coherent Systems

Another type of system classification is based on how the individual components are related to each other [Siegenfeld and Bar-Yam 2020]:

- RANDOM: Each component behaves independently of the others;
- COHERENT: Each component behaves in the same way; and
- CORRELATED: Systems which are in between random and coherent systems.

For example, the direction of passengers seated on a 14 hour airplane flight during takeoff is COHERENT, the length of time they sleep during flight is RANDOM, and their choice of in-flight entertainment is CORRELATED.

Entropy is a measure of disorder of a system. Random systems have large entropy, and perfectly coherent systems zero entropy. Complex systems may undergo phase transitions from coherent states with zero entropy to correlated/random states with positive entropy.

Emergent Global Behavior

Models can help explain how recognizable macroscopic or global states emerge from microscopic or local interactions. For example, synchronized movement (coherence) of a flock of birds can be modelled as emergent macroscopic system behavior based on a few simple local rules (Tranquillo 2019)

- SEPARATION: Birds keep a minimum distance from other birds;
- ALIGNMENT: Birds move towards the average direction of the neighbors they see around them; and
- COHESION: Birds move towards the average center of mass of the birds around them.

See also the video "Murmuration of Starling - Simulation of Flocking Boids in C++ / DirectX 11- 2" YouTube, uploaded by Gavin Wood, 6 March 2015, https://www.youtube.com/watch?v=nbbd5uby0sY. Permissions: YouTube Terms of Service

Exercises

Exercises

2.2.1 During the civil war, Tigray was suppressed into a closed system. Shortly after the war ended in November 2022, UNICEF reported over 2,000 schools were closed. Elementary school students' walk to school increased from 1 to 4 miles, and student to classroom ratios were over 400:1. Without blackboards, a teacher resorted to use of a large stone in working with students (**Figure 4**).

Figure 4 A teacher used a rock in lieu of a blackboard as a result of the ransacking/blockade of Tigray during the civil war.

How would the rebuilding of education be different within an isolated, closed, or open educational system?

2.2.2 How might coherence (or lack thereof) of IDP camps impact the complexity of a response?

2.2.3 Show that a clearly visible pattern emerges if we apply Stephen Wolfram's Rule 90 to a single black square on a large graph paper (https://mathworld.wolfram.com/SierpinskiSieve.html). The rule explains the coloring of squares on the row below the current row.

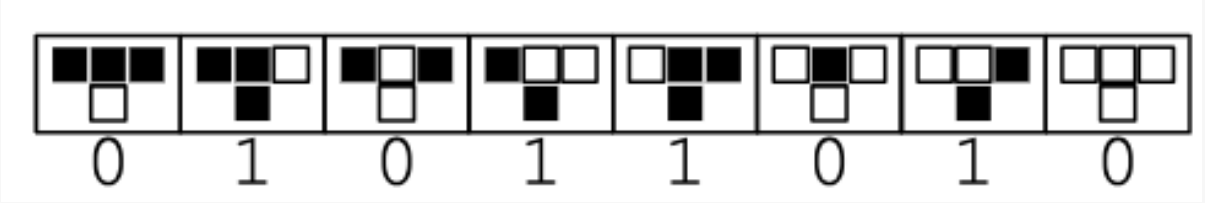


Figure 5 Wolfram's Rule 90 shows how to compute the cell in the row beneath given the three cells above it. What pattern emerges if we apply this rule to a row with a single black square?

15.2.3 Linearization

Linear systems are foundational and a starting point for the study of dynamical systems. Nonlinear systems are needed to model complexity and have been important in the study of dynamical systems for the past 50 years. In general, non-linear systems do not have exact solution methods. Non-linear systems may be approximated by linear systems near equilibrium points.

Consider a non-linear system

$$x' = f(x, y)$$

$$y' = g(x, y)$$

An equilibrium point (x^*, y^*) for the non-linear system satisfies

$$f(x^*, y^*) = 0$$

$$g(x^*, y^*) = 0$$

The Jacobian matrix (all partials are evaluated at the equilibrium point) is

$$J = \begin{pmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{pmatrix}.$$

If the real part of the eigenvalues of the Jacobian matrix are non-zero, then the behavior of the non-linear system at the equilibrium point (x^*, y^*) is given qualitatively by the behavior of the equilibrium point at (0,0) of the linear system

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

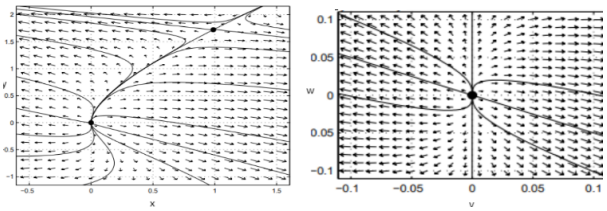


Figure 6 (Left) Nonlinear solutions (Right) Linearization at the equilibrium point (0,0)

Let A be the Jacobian matrix for a linearized 2D non-linear system at an equilibrium point $P = (x^*, y^*)$. **Figure 7** summarizes the qualitative behavior of the nonlinear system near P (Exercise 2.3.1).

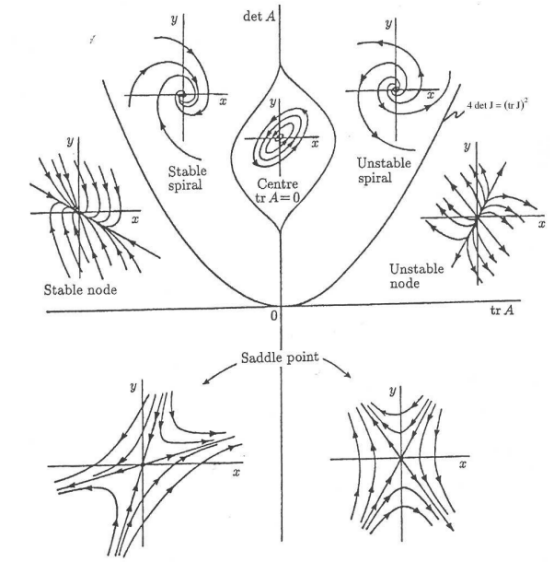


Figure 7 Classification of equilibria for linearized 2-D systems. Figure Credit: Appendix A of J.D. Murray's [2002] *Mathematical Biology*. (Used with permission of the author.)

Logistic Growth

Moving away from an equilibrium point, the need to transition from an approximating linear behavior to non-linear behavior is illustrated using COVID-19 data shown in **Figure 8**. Mathematically, let k and M be positive constants, and consider the logistic growth equation

$$\frac{dz}{dt} = kz(1 - \frac{z}{M}),$$

with $z(0) = z_0$ positive and close to 0. Note that when z/M is small, the linear exponential growth model $\frac{dz}{dt} = kz$ is a good approximation for the nonlinear logistic model. Over time, as $z \to M$, the non-linearity which arises from multiplication of kz by $(1 - \frac{z}{M})$ is needed to cap the long range growth by reducing the value of $\frac{dz}{dt} = kz(1 - \frac{z}{M})$ to zero.

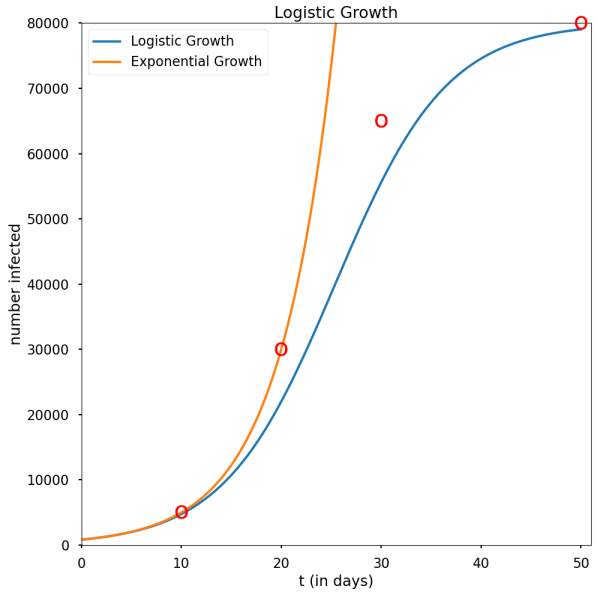


Figure 8 COVID-19 data plotted as circular points together with exponential and logistic growth curves.

For simplicity, set $k = 2$, $M = 2$ so that $\frac{dz}{dt} = g(z) = 2z - z^2$. The stability of the equilibrium when $z = 2$ can be inferred from the local minimum point of a potential function $V(z) = -z^2 + \frac{1}{3}z^3$ (**Figure 9**) which satisfies $V'(z) = -g(z)$. When $z > 2$, $V'(z) > 0 \Rightarrow \frac{dz}{dt} < 0$ (so z decreases towards the equilibrium value); and when $z < 2$, $V'(z) < 0 \Rightarrow \frac{dz}{dt} > 0$ (so z increases towards the equilibrium value. The idea that optimization of an auxiliary function (in this case the potential $V(z)$) can be used to analyze system behavior will be important in **Section 3**.

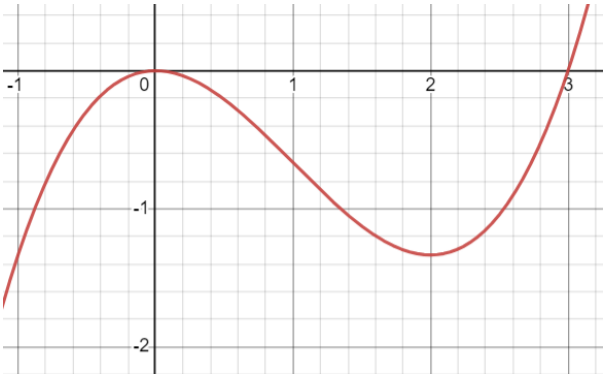


Figure 9 The minimum point on the potential function $V(z) = -z^2 + \frac{1}{3}z^3$ corresponds to the stable equilibrium $z = 2$.

Exercises

Exercises

2.3.1 Let A be the Jacobian matrix for a linearized 2D non-linear system at an equilibrium point $P = (x^*, y^*)$. Use Figure 7 to explain how eigenvalues of A can be used to find the following qualitative behavior of the nonlinear system near P.

2.3.2 Use linearization to describe the behavior of the following system near its equilibrium at (0,0):

$$\frac{dx}{dt} = 3x - y^2$$

$$\frac{dy}{dt} = \sin(y) - x$$

15.2.4 Bifurcations

It is often the case that the behavior of a physical system is dependent on a parameter value such as temperature. In some cases, there is a critical parameter value at which the macroscopic state of a system undergoes a fundamental change in behavior (for example, transition of water from liquid to solid state).

Consider a natural environment with two types of animals: a single predator (eg. foxes) and a single prey (eg. rabbits). There are different possibilities for the equilibrium states of the predator density F^* and prey density C^*:

i) the predator population dies off ($F^* = 0$) for lack of food and there is a stable number of prey ($C^* > 0$) that remains;

ii) there is a stable number of both predators and prey ($F^* > 0, C^* > 0$); or

iii) the number of predators and prey cycle periodically (see **Figure 10**).

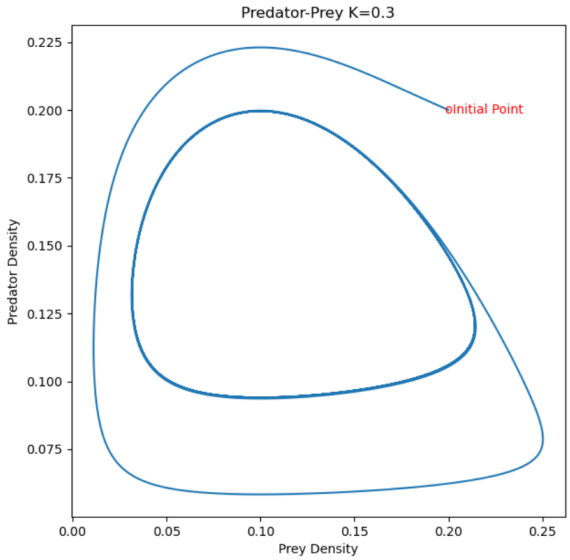


Figure 10 A predator-prey model which exhibits a stable limit cycle seen in the middle region of the diagram. The number of predator and prey both fluctuate periodically.

For this system, the carrying capacity K of the predator population is a bifurcation parameter (See **Figure 11** and Exercise 2.3.1):

a) When $K > 0$ is small, a stable steady state occurs at $(F, C) = (0, C^*)$ with $C^* > 0$.

b) For higher values of K, the equilibrium $(0, C^*)$ loses its stability and there arises a new stable equilibrium (sink) (F^*, C^*) with $F^* > 0$ and $C^* > 0$.

c) For even higher values of K, this equilibrium (F^*, C^*) becomes a stable spiral sink.

d) For still higher values of K, the equilibrium (F^*, C^*) loses its stability and a stable limit cycle (closed curve) solution appears.

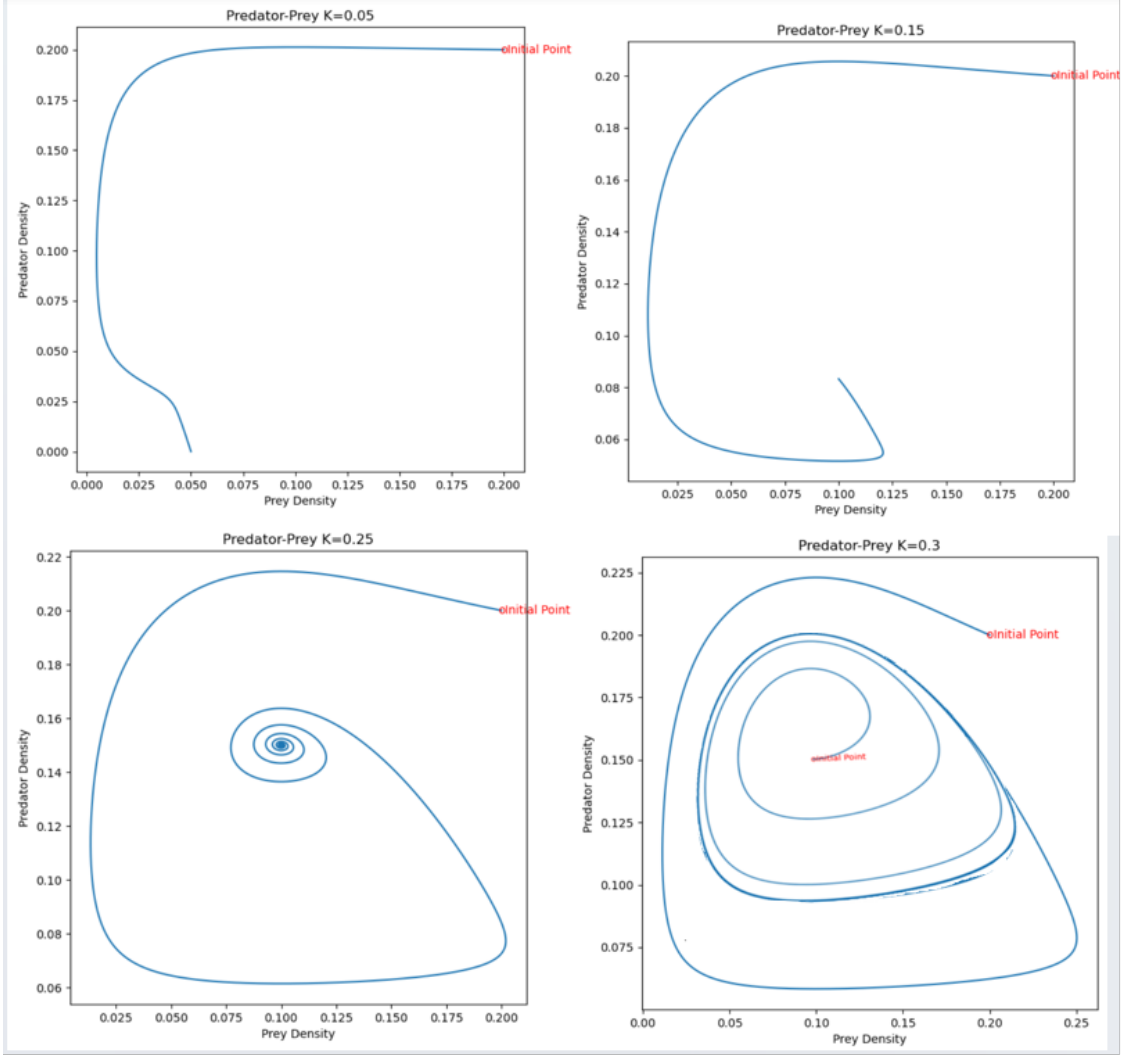


Figure 11 Bifurcation in the Predator-Prey Equations occurs as the predator carrying capacity K is varied.

Exercises

Exercises

2.4.1 Consider the Rosenzweig-MacArthur (RM) predator-prey model [Martin and de Roos 2023]

$$\frac{dF}{dt} = rF(1 - \frac{F}{K}) - \frac{aF}{1 + ahF}$$

$$\frac{dC}{dt} = \epsilon \frac{aF}{1 + afF} C - \mu C$$

where

F= prey density (eg rabbits); C= predator density (eg foxes); dF/dt = dynamic change of prey; dC/dt = dynamic change of predator; r = prey natural growth rate; K = prey carrying capacity; a = attack rate; h = handling rate (predator needs h time units to consume prey); e = prey offspring proportionality constant; μ = predator death rate.

Use a JNB to confirm that

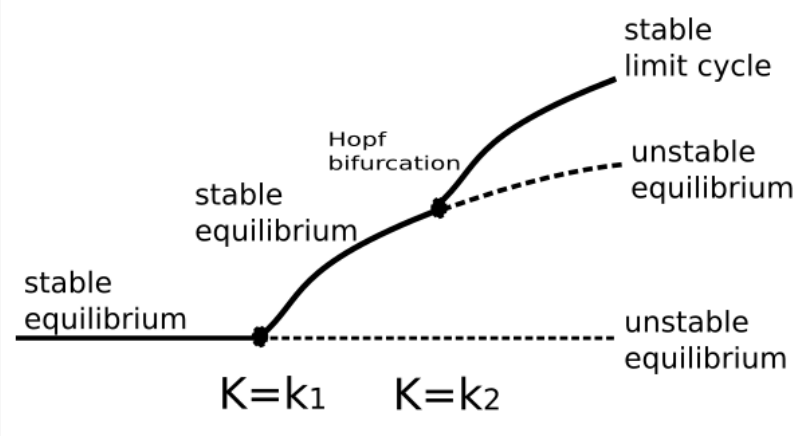
a) When K>0 is small, a stable steady state occurs at (F*,C*)=(K,0)

b) For higher values of K, there is a steady state with F*>0, C*>0.

c) For even higher values of K, the equilibrium is a spiral sink.

d) For still higher values of K, the equilibrium loses its stability and a stable limit cycle (closed curve) solution appears.

2.4.2 The RM model exhibits what is called a Hopf bifurcation, as described in the diagram below:



Explain how the Hopf bifurcation diagram corresponds to your solution to Exercise 2.4.1.

2.4.3 One dimensional maps can also undergo bifurcations. Consider the 1D map $x_{n+1} = rx_n(1 - x_n)$.

a) Find the period one point (fixed point) for this map when $r < 2$.

b) This map undergoes a period doubling bifurcation as shown in the figure below. What type of universality is suggested by this sequence of bifurcations?

Period	Bifurcation parameter (r)	Ratio $\left(\frac{r_{n-1}-r_{n-2}}{r_n-r_{n-1}}\right)$
2	3.0	–
4	3.449	–
8	3.544	4.7514
16	3.564	4.6562
32	3.658	4.6683
64	3.569	4.6686
128	3.5696	4.6692
256	3.5699	4.6694

15.2.5 Normal vs Power Law Behavior

A system which evolves over time is deterministic if its future states are modelled by a deterministic function of time. In contrast, future states may depend on a random process.

One-dimensional random walks are among the simplest type of non-deterministic systems. Let x_t denote the position at discrete times $t = 0, 1, 2, ...$ of an object moving at random on a one-dimensional integer lattice. Consider a random walk with $x_0 = 0$ and

$$x_{t+1} = x_t + e_t$$

where $e_t = \pm 1$ with equal probability. The random variables e_t and $e_{t'}$ $(t \neq t')$ are assumed to be independent and identically distributed (IID).

A single random walk is impossible to predict, and any observed trend in a specific random walk is not in general a common feature of all random walks. Yet, a large number of random walks will have a predictable expected position $< x_t >= 0$ and variance $< x_t^2 >= t$ (**Figure 14**) since

$< x_{t+1} >=< x_t > + < e_t >=< x_t >=< x_0 >= 0$ for all t; and

$< x_{t+1}^2 >=< x_t^2 > +2 < x_t >< \epsilon_t > +1 =< x_t^2 > +1 \Rightarrow < x_t^2 >=< x_0^2 > +t = t.$

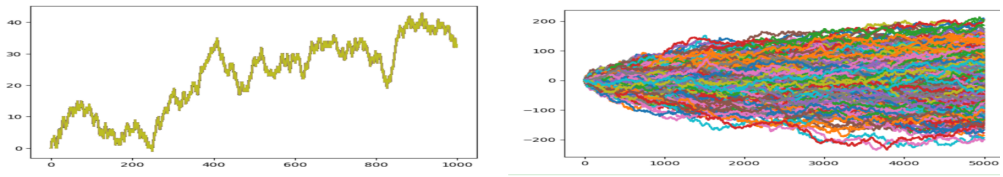


Figure 14 (Left) A single random walk is unpredictable. (Right) A large number of random walks have predictable mean $< x_t >= 0$ and variance $< x_t^2 >= t$.

We can also create a random walk using variable step sizes by random draws from a continuous probability distribution. **Figure 15** shows the difference between random walks which are normal vs. Cauchy (power-law) distributed. The former gives rise to two-dimensional Brownian motion where both the x and y step sizes are normally distributed. Very large steps sizes have negligible probability of occurrence and are not observed. On the other hand, large step sizes ('long-flights') are observed in power-law (heavy-tail) distributed random walks. These may correspond to catastrophic events (eg. a stock market crash). Complex systems are often characterized by power-law behavior [Fieguth 2021].

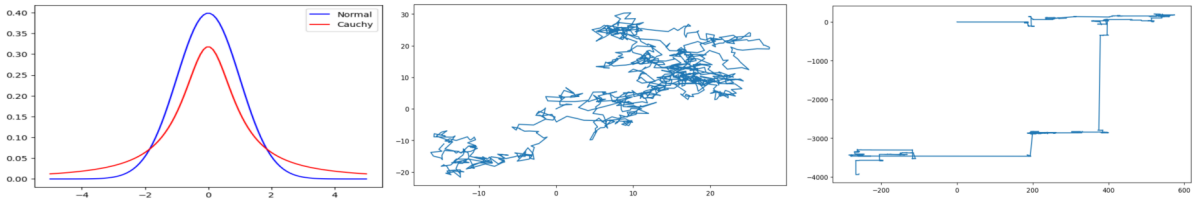


Figure 15 Left) Normal and Cauchy (power-law) distributions. (Center) Brownian motion with moves normally distributed. (Right) Heavy tail of a Cauchy distribution gives rise to long-flights in the random walk.

Exercise

Exercise

2.5 Use a JNB to re-create the graphs in **Figure 15**.

15.3 Mathematical Concepts from Equilibrium Statistical Physics

Many important concepts in the study of complex systems trace back to the highly mathematical field of statistical physics [Bertin 2022]. This section touches on just a few of these concepts for equilibrium systems:

- The Hamiltonian (total energy) of a system in equilibrium (Sec. 3.1);
- Discrete Ising spin models and entropy for systems with a specified energy (Sec 3.2); and
- Maximum likelihood and phase transitions involving a change in a system parameter such as temperature T (Sec 3.3).

The section concludes with a simplified Schelling model (Sec. 3.4) as an example how a statistical physics approach can model segregation.

15.3.1 The Hamiltonian

For systems in equilibrium, the Hamiltonian can illustrate important concepts such as conservation of energy and time reversibility.

Spring Mass Problem

An idealized spring-mass model considers a mass m attached to a spring which exerts the total force F on the mass moving in 1 dimension:

$$F = -kx.$$

Here $k > 0$ is a spring constant, and x is the distance the spring is stretched ($x > 0$) or compressed ($x < 0$) from its rest position at $x = 0$.

From Newton's 2nd Law, we know that $F = ma$, which leads to the ODE

$$mx''(t) = -kx.$$

The general solution is $x(t) = A\cos(\omega t + \phi)$, where $\omega = \sqrt{\frac{k}{m}}$. This indicates that the motion is periodic in the absence of any frictional or damping forces. .

Total Energy (Hamiltonian)

In this idealized model, the total energy of the spring-mass system is conserved, so the solution indicates that a periodic oscillation will continue for all t. Hamiltonian dynamics focuses on the system's total energy, denoted H (for Hamiltonian) or E (for energy). The total energy is the sum of kinetic energy ($\frac{1}{2}mv^2$) and potential energy U. Note that momentum $p = p(t)$ is defined as $p(t) = mv(t) = mx'(t)$ so the kinetic energy in terms of p is equal to $\frac{p^2}{2m}$. The potential function U is related to the conservative force F according to

$$U'(x) = -F(x).$$

It follows that the potential energy is $U(x) = \frac{1}{2}kx^2$.

Summing the kinetic and potential energy, gives the Hamiltonian

$$H(x,p) = \frac{p^2}{2m} + \frac{1}{2}kx^2.$$

Properties of the 1D Hamiltonian Dynamics

Suppose our starting point for analyzing the 1D dynamics is to give the total energy in terms of the Hamiltonian $H(x,p) = \frac{p^2}{2m} + \frac{1}{2}kx^2$. We could then show the following hold:

(Dynamical Equations) The state vector consisting of position and momentum $< x(t), p(t) >$, satisfies (Exercise 3.1.1)

$$\frac{dx}{dt} = \frac{\partial H}{\partial p}$$

$$\frac{dp}{dt} = -\frac{\partial H}{\partial x}.$$

(Conservation of Energy) This can be proven (Exercise 3.1.2) by showing that if $E = H(x,p)$, then

$$\frac{dE}{dt} = 0.$$

Note that as a consequence, if we let

$$X = \sqrt{\frac{k}{2}}x, \qquad Y = \frac{p}{\sqrt{2m}}$$

then in (X,Y) *phase space*, the particle's trajectory is along a circle

$$X^2 + Y^2 = E.$$

(Time Reversibility) Time reversibility means that the dynamical equations for the time interval $(0 \le t \le t_0)$ are invariant under the transformations

$$t \to t^* = t_0 - t$$

$$x = x(t) \to x^* = x(t^*)$$

$$v = \frac{dx(t)}{dt} \to v^* = \frac{dx^*}{dt^*} = -\frac{dx^*}{dt} = -\frac{dx(t^*)}{dt} = -v(t^*)$$

$$p = p(t) \to p^* = mv^* = -mv(t^*)$$

The Hamiltonian $H(x^*,p^*) = \frac{(p^*)^2}{2m} + \frac{1}{2}k(x^*)^2$ for time reversal specifies the dynamical equations (Exercise 3.1.3)

$$\frac{dx^*}{dt^*} = \frac{\partial H}{\partial p^*}$$

$$\frac{dp^*}{dt^*} = -\frac{\partial H}{\partial x^*}$$

That is, the time-reversed state vector $< x^*, p^* >$ is also a feasible dynamical trajectory.

The Hamiltonian for a single particle in 1D can be generalized to describe the dynamics of N particles in 3D space. Conservation of energy and time reversibility continue to hold. Moreover, we state:

FUNDAMENTAL POSTULATE OF EQUILIBRIUM STATISTICAL PHYSICS

In equilibrium, the total energy E is conserved, and all configurations with energy E are uniformly probable.

Exercises

Exercises

These problems are based on the idealized 1D spring-model whose dynamics are given by the ODE

$$mx''(t) = -kx.$$

with general solution $x(t) = A\cos(\omega t + \phi)$, where $\omega = \sqrt{\frac{k}{m}}$.

3.1.1 Show that

$$\frac{dx}{dt} = \frac{\partial H}{\partial p}$$

$$\frac{dp}{dt} = -\frac{\partial H}{\partial x}$$

where H is the Hamiltonian given by $H(x,p) = \frac{p^2}{2m} + \frac{1}{2}kx^2$.

3.1.2 Use a chain rule for 2 variables to show that if $E = H(x,p)$, then $\frac{dE}{dt} = 0$.

3.1.3 Show that under the the transformations

$$t \to t^* = t_0 - t$$

$$x = x(t) \to x^* = x(t^*)$$

$$v = \frac{dx(t)}{dt} \to v^* = \frac{dx^*}{dt^*} = -\frac{dx^*}{dt} = -\frac{dx(t^*)}{dt} = -v(t^*)$$

$$p = p(t) \to p^* = mv^* = -mv(t^*)$$

the Hamiltonian $H(x^*, p^*) = \frac{(p^*)^2}{2m} + \frac{1}{2}k(x^*)^2$ specifies the dynamical equations

$$\frac{dx^*}{dt^*} = \frac{\partial H}{\partial p^*}$$

$$\frac{dp^*}{dt^*} = -\frac{\partial H}{\partial x^*}$$

That is, the time-reversed state vector $< x^*, p^* >$ is also a possible dynamical trajectory.

15.3.2 Ising Spin Models and Entropy

Ensembles of a finite number of particles can be classified according to whether they exchange energy or mass with their environment.

Microcanonical Ensemble: An isolated system $\mathcal{S}$ which exchanges neither energy nor mass with its environment $\mathcal{R}$.

Canonical Ensemble: A system $\mathcal{S}$ which exchanges energy with its environment $\mathcal{R}$. The total system $\mathcal{S} \cup \mathcal{R}$ is assumed to be isolated.

Grand-canonical Ensemble: A system $\mathcal{S}$ which exchanges both mass and energy with its environment $\mathcal{R}$. The total system $\mathcal{S} \cup \mathcal{R}$ is assumed to be isolated.

The Ising model is a simple discrete model which can be used to study energy and mass exchange of a system with its environment, as well as phase transitions. For our discrete model, we restrict particle positions to a lattice of evenly spaced points in dimension D=1,2, …. For example, the points on a number line representing the integers $\mathbb{Z}$ form a 1D lattice. Similarly, the points with integer coordinates $\{(m, n) \mid m, n \in \mathbb{Z}\}$ forms a 2D lattice. A system of N particles may be confined to a certain portion of the lattice by a boundary, and exchange energy or mass with its environment (lattice points outside the boundary which may be occupied by other particles). Time is incremented in discrete time steps $t_0, t_1, t_2, ...$

A **spin system** assigns a value of +1 or -1 to each particle. The Hamiltonian, or total energy, E, of the system is computed using the spin values. In the basic Ising model for an N particle system,

$$E = -J \sum_{i,j} s_i s_j - h \sum_{i=1}^{N} s_i$$

where

the values of i, j in the first sum run over a specified set of lattice points such as "nearest neighbors." For example, in dimension $D = 1$, a particle has 2 nearest neighbors, and in $D = 2$, there are 4 nearest neighbors.

$J \geq 0$ is called the *spin coupling constant*. ($J = 0$ if the spins are independent of each other, i.e. totally decoupled).

$h \geq 0$ is the external magnetic field constant and the magnetization M is equal to

$$M = \sum_{i=1}^{N} s_i \qquad (s_i = \pm 1)$$

a state or configuration C is specified by a complete list of spin values $(s_1, ..., s_N)$

$\Omega(E)$ is the number of configurations with energy E

if E is the energy of the system in equilibrium, all spin configurations with energy E have equal positive probability of occurrence and all other configurations have zero probability of occurrence.

Isolated Paramagnetic Spin Model

For the case of independent spins, we set $J = 0$, and the energy E simplifies to

$$E = -h \sum_{i=1}^{N} s_i = -hM, \qquad (s_i = \pm 1),$$

where $M = \sum_{i=1}^{N} s_i$ is the magnetization.

Given the energy E, let N_+ be the number of particles with spin +1 and N_- be the number of particles with spin -1. Using algebra, we can show (Exercise 3.2.1) that

$$N_+ = \frac{1}{2}(N - \frac{E}{h})$$

and

$$N_- = \frac{1}{2}(N + \frac{E}{h}).$$

It follows that

$$\Omega(E) = \frac{N!}{N_+!N_-!}$$

The entropy $\mathbf{S}(E)$ is computed as

$$\mathbf{S}(E) = \ln \Omega(E).$$

Using Stirling's formula

$$\ln N! \approx N \ln N - N,$$

the entropy **S** of the parametric spin model of a system with energy E is (Exercise 3.2.2)

$$\mathbf{S}(E) \approx \ln \frac{N^N}{N_-^{N_-} N_+^{N_+}}.$$

Canonical Ensembles

We now consider a canonical equilibrium system $S_{tot} = \mathcal{S} \cup \mathcal{R}$ where system $\mathcal{S}$ exchanges energy with its environment $\mathcal{R}$. The total system $S_{tot} = \mathcal{S} \cup \mathcal{R}$ is considered to be a closed system.

Total Energy: The total energy $E_{tot} = E_{\mathcal{S}} + E_{\mathcal{R}}$ is constant although energy can be exchanged between $\mathcal{S}$ and $\mathcal{R}$.

System Configurations (States): A total system configuration (state) C_{tot} is written as $C_{tot} = (C_{\mathcal{S}}, C_{\mathcal{R})}$ where $C_{\mathcal{S}}$ is a configuration of $\mathcal{S}$ and $C_{\mathcal{R}}$ is a configuration of $\mathcal{R}$.

Independence: $\Omega_{tot}(E_{\mathcal{S}} \mid E_{tot}) = \Omega_{\mathcal{S}}(E_{\mathcal{S}})\Omega_{\mathcal{R}}(E_{tot} - E_{\mathcal{S}}) = \Omega_{\mathcal{S}}(E_{\mathcal{S}})\Omega_{\mathcal{R}}(E_{\mathcal{R}})$. (Given the total energy E_{tot}, for a system energy $E_{\mathcal{S}}$, the admissible configurations of $\mathcal{S}$ and $\mathcal{R}$ are considered to be statistically independent.)

Under maximum likelihood, the most probable value $E_{\mathcal{S}}^*$ of $E_{\mathcal{S}}$ satisfies

$$\frac{\partial}{\partial E_{\mathcal{S}}} \mid_{E_{\mathcal{S}}^*} \ln \Omega_{tot}(E_{\mathcal{S}} \mid E_{tot}) = 0.$$

From this it follows that (Exercise 3.2.3)

$$\frac{\partial \ln \Omega_{\mathcal{S}}}{\partial E_{\mathcal{S}}} \mid_{E_{\mathcal{S}^*}} = \frac{\partial \ln \Omega_{\mathcal{R}}}{\partial E_{\mathcal{R}}} \mid_{E_{\mathcal{R}^*}}$$

where $E_{\mathcal{S}^*} + E_{\mathcal{R}^*} = E_{tot}$. The common value of the partial derivatives at the maximum likelihood energy is denoted β and called the statistical temperature. For large systems, it can be shown that [Bertin 2021]

$$\beta = \frac{\partial \mathbf{S}_{tot}}{\partial E_{tot}}$$

where $\mathbf{S}_{tot} = \ln \Omega_{tot}$ is the entropy of the total system. A law of thermodynamics states that for physical systems $dE_{tot} = T d\mathbf{S}_{tot}$ where T is the usual thermodynamic temperature. Hence, $\beta = 1/T$.

Grand Canonical Ensembles

For a grand-canonical ensemble, the macroscopic system $\mathcal{S}$ exchanges both energy and particles with the environment $\mathcal{R}$. The modeling of GC ensembles is a natural extension of the modeling of canonical ensembles.

Total energy: $E_{tot} = E_{\mathcal{S}} + E_{\mathcal{R}}$

Total number of particles: $N_{tot} = N_{\mathcal{S}} + N_{\mathcal{R}}$

Probability of Configuration: $C_{\mathcal{S}}$:

$$P_{GC}(C_{\mathcal{S}}) = K\Omega_{\mathcal{R}}(E_{tot} - E_{\mathcal{S}}(C_{\mathcal{S}}), N_{tot} - N_{\mathcal{S}}(C_{\mathcal{S}}) = Kexp[\mathbf{S}_{\mathcal{R}}(E_{tot} - E_{\mathcal{S}}, N_{tot} - N_{\mathcal{S}}(C_{\mathcal{S}})]$$

where $K = 1/Z_{GC}$ is a normalization constant.

The linear approximation of $A = \mathbf{S}_{\mathcal{R}}(E_{tot} - E_{\mathcal{S}}, N_{tot} - N_{\mathcal{S}})$ is

$$A = \mathbf{S}_{\mathcal{R}}(E_{tot}, N_{tot}) - \beta E(C_{\mathcal{S}}) + \frac{\mu}{T} N_{\mathcal{S}}$$

where $\beta = 1/T$ is the statistical temperature and $\mu = -T\frac{\partial \mathcal{S}_{\mathcal{R}}}{\partial N_{\mathcal{R}}}$ is called the chemical potential.

The grand-canonical partition function is defined as

$$Z_{GC} = \sum_{C_{\mathcal{S}}} exp[-\frac{1}{T}E(C_{\mathcal{S}}) + \frac{\mu}{T}N_{\mathcal{S}}(C_{\mathcal{S}})].$$

In terms of the partition function, the probability of configuration $C_{\mathcal{S}}$ (called the grand-canonical distribution) is

$$P_{GC}(C_{\mathcal{S}}) = \frac{1}{Z_{GC}} exp[-\frac{1}{T}E_{\mathcal{S}(C_{\mathcal{S}})+\frac{\mu}{T}N_{\mathcal{S}}(C_{\mathcal{S}})]}$$

Exercises

Exercises

3.2.1 Show that the following hold for the paramagnetic spin model with

$$E = -h\sum_{i=1}^{N} s_i, \qquad (s_i = \pm 1,)$$

$$N_+ = \frac{1}{2}(N - \frac{E}{h})$$

and

$$N_- = \frac{1}{2}(N + \frac{E}{h}).$$

3.2.2 Use Stirling's approximation to show that the entropy for the paramagnetic spin model of a system with energy E is given by

$$\mathcal{S}(E) \approx \ln \frac{N^N}{N_-^{N_-} N_+^{N_+}}.$$

3.2.3 Under maximum likelihood, the most probable value $E_{\mathcal{S}}^*$ of $E_{\mathcal{S}}$ satisfies

$$\frac{\partial}{\partial E_{\mathcal{S}}} |_{E_{\mathcal{S}}^*} \ln \Omega_{tot}(E_{\mathcal{S}} \mid E_{tot}) = 0.$$

Show that this implies

$$\frac{\partial \ln \Omega_{\mathcal{S}}}{\partial E_{\mathcal{S}}} \mid_{E_{\mathcal{S}^*}} = \frac{\partial \ln \Omega_{\mathcal{R}}}{\partial E_{\mathcal{R}}} \mid_{E_{\mathcal{S}^*}}$$

3.2.4 Explain the intuition behind the formula for the grand-canonical distribution:

$$P_{GC}(C_{\mathcal{S}}) = \frac{1}{Z_{GC}} exp[-\frac{1}{T} E_{\mathcal{S}(C_{\mathcal{S}}) + \frac{\mu}{T} N_{\mathcal{S}}(C_{\mathcal{S}})]}$$

3.2.5 In information theory, Shannon entropy is defined as

$$\mathbf{S} = -\sum_{C} P(C) \ln P(C).$$

Show that

a) For a microcanonical ensemble with energy $E_{\mathcal{S}}$, the Shannon entropy $\mathbf{S_{mic}} = -\sum_{C_{\mathcal{S}}} P(C_{\mathcal{S}}) \ln P(C_{\mathcal{S}})$ is given by

$$\mathbf{S_{mic}} = \ln \Omega(E).$$

b) For a canonical ensemble, recalling that $P_{\mathcal{S}}(C_{\mathcal{S}}) = \frac{1}{Z} e^{-E(C_{\mathcal{S}})/T}$ and $Z = \sum_{C_{\mathcal{S}}} e^{-E_{\mathcal{S}}(C_{\mathcal{S}})/T}$, the Shannon entropy $\mathbf{S_{can}}$ is

$$\mathbf{S_{can}} = \ln Z + \beta < E(C_{\mathcal{S}}) > .$$

15.3.3 Maximum Likelihood and Phase Transitions

Phase transitions are the sudden onset of a macroscopic phenomenon when a system parameter crosses a critical value. For example, magnetization occurs below a critical temperature in the Ising model. We are familiar with time reversible phase transitions such as water to ice and ice to water. (In more complex dynamics, catastrophic (irreversible) system bifurcations may occur at critical parameter levels.) We give an example of a phase transition using the Ising model in fully connected geometry.

Fully Connected Ising Model

Consider the Hamiltonian

$$E_R = -\frac{J}{N} \sum_{i<j} s_i s_j + E_0.$$

Note that the spins in the summation are fully connected rather than nearest-neighbors. The factor of N is therefore needed in the denominator to keep the energy per spin from diverging to infinity as $N \to \infty$.

The energy can be expressed in terms of the magnetization $M = \sum_{i=1}^{N} s_i$ as (Exercise 3.3.1)

$$E_R = -\frac{J}{2} N m^2,$$

where $m = M/N$ is the magnetization per spin and $E_0 = -J/2$.

The distribution of the magnetization per spin m is given by

$$P(m) = \frac{1}{Z} \sum_{C:m(C)=m} e^{-\beta E(C)} = \frac{1}{Z} e^{\mathbf{S}(m) + \frac{1}{2}\beta N m^2}$$

where the number of configurations with magnetization m is

$$\Omega(m) = e^{\mathbf{S(m)}} = \frac{N!}{N_+!N_-!}.$$

$$N_+ = \frac{N}{2}(1+m)$$

and

$$N_- = \frac{N}{2}(1-m).$$

One can show that [Bertin, 2021]

$$P(m) = e^{-Nf(m)}$$

where

$$f(m) = f_0(T) + \frac{1}{2}(1-\frac{J}{T})m^2 + \frac{1}{12}m^4 + O(m^6)$$

and $f_0(T)$ is a temperature dependent constant to ensure $f(m) \geq 0$.

The following phase transition occurs (Exercise 3.3.2):

No Magnetization If $T \geq T_{crit}$, $f(m)$ has a single minimum point that occurs when $m = 0$. (no magnetization occurs)

Magnetization If $T < T_{crit}$, $f(m)$ has two symmetric minimums at $\pm m_0$. (positive/negative magnetization occurs)

Exercises

Exercises

3.3.1 Show that for the fully connected Ising model,

$$E_R = -\frac{J}{2}Nm^2,$$

where $m = M/N$ is the magnetization per spin.

3.3.2 Consider the function $f_T(m) = \frac{1}{2}(1-\frac{1}{T})m^2 + \frac{1}{12}m^4$. Show that there exists a critical value T_{crit} such that $f_T(x)$ has a single absolute min at m=0 when $T \geq T_{crit}$ and symmetric absolute min at $m = \pm m_0$ when $T < T_{crit}$.

15.3.4 Simplified Schelling Model and Segregation

The Schelling model is an example of how the ideas of statistical physics might be applied within urban science. The Schelling model represents dynamics of residential moves within a city. The city is modelled as a checkerboard divided into cells. Two types of agents reside in the city, with at most one agent in each cell. A utility function u describes the degree of satisfaction of each agent with the neighborhood in which they reside and governs the dynamics of agent moves. Segregation consists of areas with higher densities of one type of agent.

A simplified Schelling model considers only one type of agent and considers segregation in the form of areas which have higher densities of agents (as opposed to a homogeneous density throughout the city). In the simplified model,

Blocks: A city is divided into a large number Q of blocks.

Cells: Each block contains H cells.

Configurations: A macroscopic configuration C of the city consists of a knowledge of the state (empty or occupied) of each cell.

Agents: Each cell contains at most one agent, so the number of agents n_q in a given block q (q=1,2,...,Q) satisfies $n_q \leq H$.

Agent Density: The density of agents is $\rho_q = n_q/H$

Utility: Each agent has the same utility function $u(\rho_q)$ which indicates the degree of satisfaction with the block it lives in.

The discrete-time movement of the agents in the city follow these rules:

Agent Movement: Agents can only move from one block to a different block.

Random Selection: At each time step, one agent and one empty cell in a different block are selected at random.

Movement Probability: The probability that the agent moves to the empty cell is

$$W(C' \mid C) = \frac{1}{1 + e^{-\Delta u/T}}$$

where C and C' denote the respective configurations before and after the move.

Change in Utility: Δu is the change in utility in moving to the empty cell.

Additional Factors: The parameter T (analogous to temperature) takes into account factors such as presence of services, shops, friends etc. that influence the decision to move.

Individual Preference: The model is individualistic in that the probability of moving only depends on the agent's change in utility. It does not consider the potential impact on other agents.

Balance Equation: The balance equation is

$$W(C' \mid C)P(C) = W(C \mid C')P(C')$$

with

$$P(C) = \frac{1}{Z} e^{-F(C)/T}$$

where Z is the analog to the partition function and

$$F(C) = -\sum_q \sum_{m=0}^{n_q} u(\frac{m}{H}).$$

To study segregation, in equilibrium, the probability distribution of the block densities (with the sum of the densities held constant) has the form

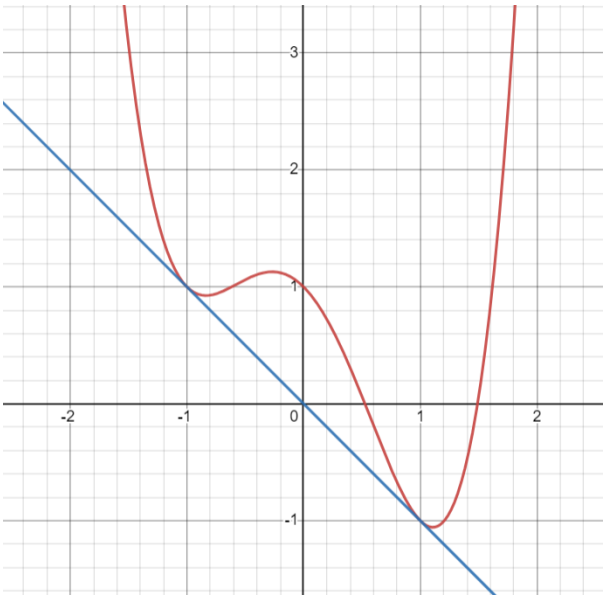
$$P(\rho_1, ..., \rho_Q) = Kexp(-\frac{H}{T} \sum_{q=1}^{Q} f(\rho_q))$$

with $H, T > 0$.

A homogeneous (non-segregated) density ρ_0 is unstable if there exist two densities ρ_1 and ρ_2 such that

$$\gamma f(\rho_1) + (1 - \gamma) f(\rho_2) < f(\rho_0).$$

The parameter γ $(0 < \gamma < 1)$ corresponds to the fraction of blocks that would have a density ρ_1 in the segregated state. This condition means that the value of the potential Φ is lower for the segregated state than the homogeneous state (see Exercise 3.4.1). Such densities can be located geometrically as points of bitangency [Bertin 2021] (see the Figure below and Exercise 3.4.2).



Exercises

Exercises

3.4.1. Show that in equilibrium the maximum likelihood configurations of densities $(\rho_1, ..., \rho_Q)$ minimizes the potential function ('free energy') $\Phi(\rho_1, ..., \rho_Q)$ defined as

$$\Phi(\rho_1, ..., \rho_Q) = \sum_{q=1}^{Q} f(\rho_q)$$

3.4.2. Find a line which is tangent to the quartic $y = f(x) = x^4 - 2x^2 - x + 1$ at two different points $P = (p, f(p))$ and $Q = (q, f(q))$, Sketch the quartic and bi-tangent line.

15.4 Societal Applications

Perhaps the most obvious example of complex social systems and justification for urban science is the proliferation of large metropolitan cities.

- RAPID URBANIZATION: The vast majority of those living a traditional subsistent existence are projected to move to urban environments. In 2007, the world's urban population surpassed the rural, and in 2021, the latter peaked and is now in decline [Bettencourt 2021]. The emergence of major metropolitan areas has shifted inter-connectivity of people from familiar local relations to global business between strangers.
- TRANSFORMATIVE POWER: Urban environments have enormous transformative power which compels a scientific understanding of how cities grow and impact their environment. There is a positive correlation between urbanization and economic and human development [Bettencourt 2021]. Informed/uninformed political decisions influencing large, high-density masses of people may support or destroy civilization.

In this section, we give three examples of urban data analysis which draw on concepts introduced earlier in this chapter. A fourth example returns to consideration of a complex problem in post-war Tigray, Ethiopia; namely, the plight of internally displaced people (IDP).

15.4.1 Incarceration and Parole

The penal population is large and complex, and every individual in this population is different. The state of each offender (incarcerated, parole, not incarcerated, not on parole) is important and must be tracked. The macroscopic state (percentage in each category) while in flux, may be expected to approach a steady state in the absence of policy changes. Raphael [2011] describes a dynamical system categorizing the U.S. population into one of three states: (a) Not Incarcerated, Not on Parole, (b) Incarcerated, (c) Parole. Table 1 gives the probability that an individual will transition from one state to another in 1980 and also in 2005. Determining the steady state populations associated with these transition probabilities is one way to assess improvement or deterioration of the criminal justice system over this 25 year period.

To this end, in 1980, the probability that someone on Parole (third row) transitioned to Not Incarcerated, Not on Parole was roughly .40; the probability that someone on Parole transitioned to being Incarcerated was .13; and the probability that a person on Parole remained on Parole was .47. Notice that the sum .40+.13+.47 =1 since the three states are the only possible states to which a parolee can transition. Each of the rows in the transition matrices must add to 1 for the same reason.

Table 1. Comparison of three-state transition probability matrix for 1980 and 2005.

	Destination state		
Origin State	Not Incarcerated, not on parole	Incarcerated	Parole
	A : 1980		
Not Incarcerated, not on parole	0.99937	.00063	0
Incarcerated	0.08211	0.52830	0.38958
Parole	0.40390	0.13073	0.46538
	B : 2005		
Not incarcerated, not on parole	0.99826	0.00174	0
Incarcerated	0.12697	0.50629	0.36674
Parole	0.29738	0.29335	0.40927

Transition matrices can be used to compute the equilibrium levels for each of the 3 groups (Not Incarcerated, Not on Parole, Incarcerated, Parole). The steady states are the proportion of the total population comprised by each group.

We now describe mathematically how the steady state incarcerated is computed for 1980 using a simplified transition probability matrix based on section A of Table 1. Consider the transition matrix M defined as follows:

$$\begin{pmatrix} .999 & .08 & .4 \\ .001 & .53 & .13 \\ 0 & .39 & .47 \end{pmatrix}$$

Note that this is the transpose of the entries in Table 1.

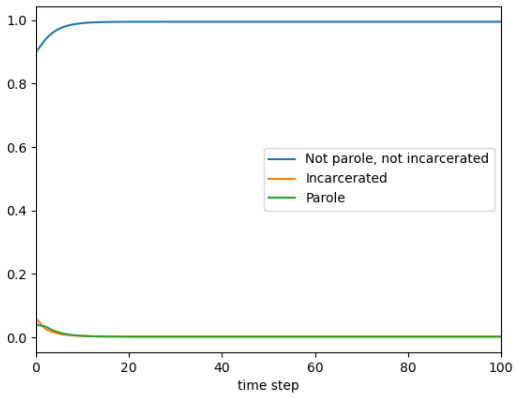
Let the steady state population levels be x_1 (Not incarcerated, Not on parole), x_2 (Incarcerated), x_3 (Parole). Then x_1, x_2, x_3 as steady state values must by definition satisfy $x_1 + x_2 + x_3 = 1$ and also

$$\begin{pmatrix} .999 & .08 & .4 \\ .001 & .53 & .13 \\ 0 & .39 & .47 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}.$$

Thus the steady state is an eigenvector of the transition matrix M with eigenvalue 1.

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} .995 \\ .003 \\ .002 \end{pmatrix}$$

Since $x_1^* = .995$, $x_2^* = .003$, and $x_3^* = .002$ the Not Incarcerated, Not on Parole population steady state proportion is 99.5% of the total population. The steady state Incarcerated population is .3% and the Parole population is .2%. An equilibrium is stable if nearby states converge to the equilibrium state as time progresses. Otherwise, the equilibrium is unstable. In this case, the graph shown below suggests these equilibrium values are stable.



15.4.2 Exercise

Exercise

4.1 Find the steady state values for 2005 and comment on its similarities/ differences with the steady state values for 1980.

15.4.3 Urban Productivity

A city's productivity is a basic socio-economic indicator often measured by GDP per capita. Such an indicator follows a power scaling law of the form

$$GDP_i = aN_i^s$$

where a and s are constants and N_i is the population of city i. Note that given GDP and population data for N cities, the scaling law implies that

$$\ln GDP_i = A + s \ln N_i.$$

with $A = \ln a$. In other words, the power s may be obtained by OLS regression of data points $(\ln N_i, \ln GDP_i)$ $(i = 1, ..., N)$ as shown in **Figure 18**.

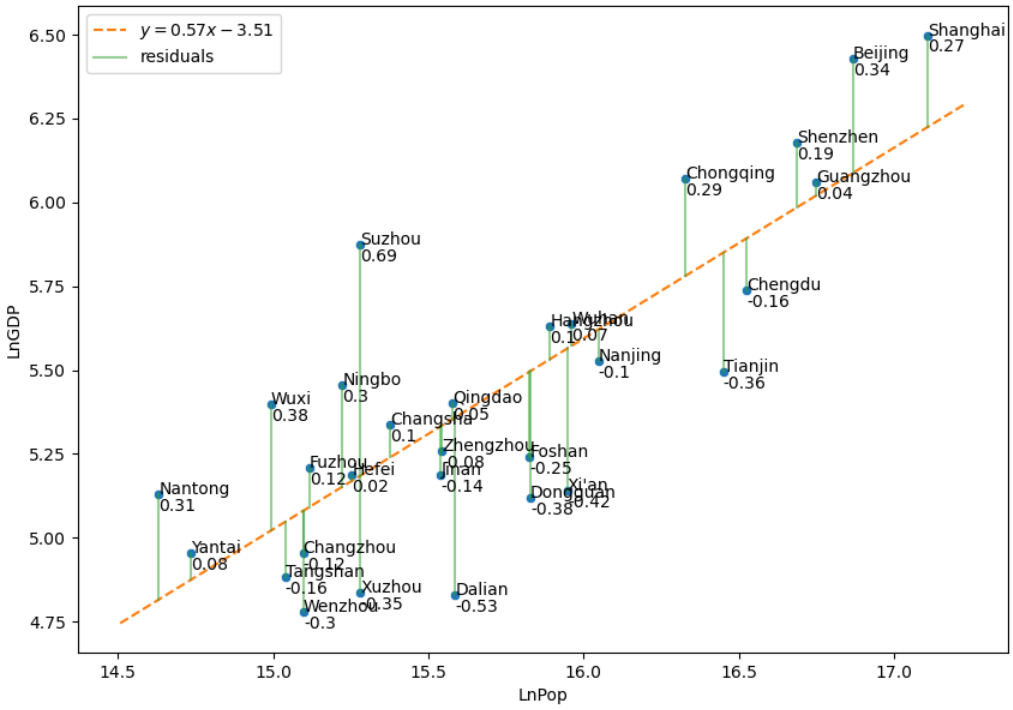


Figure 18 SAMI for GDP of the largest cities in China shows the scaling behavior as a regression line and uses residuals to assess city productivity.

SAMI (Scale Adjusted Metropolitan Index) uses the residuals (deviation from the expected power law scaling) rather than the (log of the) GDP values to assess city performance. One notes that Suzhou has an exceptionally large residual. In fact, Suzhou experimented with a central business park patterned after Singapore and became one of the most highly developed and prosperous cities in China.

Exercise

Exercise

4.2 Where would Hong Kong fall if added to the SAMI graph (**Figure 18**)?

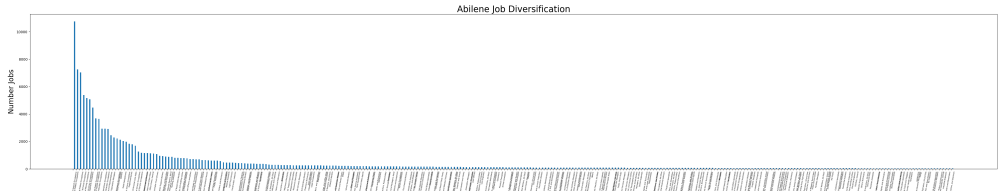
15.4.4 Job Diversification

Another interesting and important question about cities is their degree of business diversification. According to US Bureau of Labor Statistics data (https://www.bls.gov/oes/tables.htm) Abilene, TX has over 70,000 jobs which can be classified into 290 types. Using data for a large number of cities, one can look for a scaling law of the form

$$D_S(N) = cN^s.$$

where $D_S(N)$ is the expected number of different job types for a city of size N. (Note that the value of $D_S(N)$ depends on the resolution of job classification.) Then as before, one can use SAMI to assess the amount of job diversification for a given city.

Business types have various classifications, including the North American Industry Classification System (NAICS), a hierarchical taxonomy (resolution levels 2-6 with 2 a broad sector such as 71 arts and entertainment and 6 the most specific classification such as 711110 dinner theater) of business types. The histogram in the figure below gives the number of jobs in each job type. Abilene TX has over 70,000 jobs classified into 290 types. (Data Source: US Bureau of Labor Statistics https://www.bls.gov/oes/tables.htm}



Data for this histogram specifies empirical probabilities that a job is of a given type. Thus, the types can be ranked in terms of decreasing empirical probabilities. The Shannon entropy $H = -\sum_{i=1}^{D_S} P(i \mid N) \ln P(i \mid N)$ where $D_S = 290$ is thus a more sophisticated measure of diversification.

Exercise

Exercises

4.3 Consider the following distributions of students turning in late assignments for three different classes:

Class	Number Students	Number Late Assignments			
		0	1	2	3
C1	30	30	0	0	0
C2	20	19	1	0	0
C3	25	15	5	2	3

a) For each of the three class distributions, compute the Shannon entropy

$$H = -\sum_{i=0}^{4} P(i) \ln P(i)$$

where $P(i)$ denotes the empirical probability that a student turns in i late assignments.

b) In this context, why does higher entropy indicate higher disorder?

15.4.5 A Simple IDP Response Model

Between Nov 2020 and Nov 2022, the Tigray region in northern Ethiopia suffered a horrific civil war. At one point their were an estimated 2.5 million internally displaced people (IDP) out of a population of 6-7 million. Even a year after the Pretoria Peace agreement, fighting by tribal militia continued in western Tigray, leaving 1 million IDPs. If the IDP problem were the only problem, the situation would be complex. The instability caused by the on-going fighting, widespread trauma due to a huge number of civilian deaths and rape victims, a crippled infrastructure in multiple sectors (eg. health, education) left leaders wondering where to begin the reconstruction.

IDP Data

We considered the data set DTM Ethiopia - SA - Tigray - R33XLSX (1.4M) (downloaded on 12/18/23 from https://data.humdata.org/dataset/ethiopia-displacement-northern-region-tigray-idps-site-assessment-iom-dtm?). This data was published by the U.N.'s International Office of Migration (IOM), gives information about IDP camps collected in Summer 2023, and was last modified on August 11, 2023 (Round 33). The dataset contains 638 rows and 468 columns, and thus has information for more than 600 IDP camps. **Figure 20** summarizes the 468 columns of information provided.

Category	Subcategory
Communication	News sources: QF-QU Mobile network/phones: QU-QV Travel: QW-QZ
Demographics	BJ-CE
Displacement	Initial Cause: CF-DD Prevented Return: DE-EF
Education	Boy attendance barriers: OI-PC Girl attendance barriers: NM-OH Pre-primary: MF-MJ Primary: MK-MO Quality/satisfaction: MZ-NL Resources: MV-MY Secondary: MP-MT Teachers: MY, NG, NH
Food	Distribution Sites: IL-IZ Fuel: GB-GC Markets: GD-GE, JA-JL, MW School Programs: MX, NP, OL
Hygiene	Hand-washing: IC-IK Women's needs: NW
Healthcare	Location and services: LP-MA Malnutrition Treatment: JQ-JR Mobile Healthcare Teams: JZ-KA Mosquito net: AV, FS, ME MUAC Screening: JM-JP Problems in health: KC-KH Problems in healthcare access: KI-LO School Programs: MX, NP, OL TSFP (Targeted Supplementary Feeding): JS-JY, KB Women's needs: JM, JU, MB-MD, NW
Latrines	Number/Quality: HG-HM, MU Problems: HN-IB Schools: MU, OD, OY
Livelihood	Access to income: PD-PF Farming: QD-QE Livestock: PG-PT, QB-QC Loss of resources: PU Programs: PV-PZ
Shelters	Basic Information: EG-EQ Materials: ER-GA
Site	2 Largest IDP needs: AS-BI 2 Largest IDPs: AK-BI Basic Information: A-X Support Management: Y-AJ
Water	Cooking, Hygiene: GO Distribution: GH-GI Drinking Water: GL-GN, MJ, MO, MT, OE, OZ Hygiene: IC-IJ Problems: GP-HF Treatment: GJ-GK Water Sources: GF-GG

Figure 20 International Office of Migration (IOM) IDP data with Excel sheet column references.

Simplifying Assumptions on IDP System Complexity

The complexity of the IDP situation is reduced by two simplifying assumptions:

- Data Reliability: the IOM data set for over 600 IDP camps is regarded as accurate. For example, number of IDPs in a camp, age and gender breakdown, and classification of a camp's overcrowding are not questioned.
- Static Equilibrium: We also assume that the 600+ IDP camps scattered across most of Tigray are stable (neither movement of IDP from one camp to another, nor formation of new camps or closing of existing camps.)

A model's output would therefore only have a certain temporal validity and would need to be re-run on future updates to the IOM data.

A Basic Response Model

With such overwhelming needs and very few NGOs responding due to the continued instability and travel warnings, we first considered the following practical assignment problem: to which IDP camp should a given NGO respond? We used the following two criteria as the basis for assignments:

Effectiveness: The first criteria we considered is effectiveness of a response. That is, an NGO should only be assigned to an IDP camp for which it has sufficient resources to meet the need.

Child Vulnerability: The second criteria we considered is child protection: what proportion of the camp population is children under the age of 3?

We first designate the need categories under consideration $1, 2, ..., k$. For example 1 might be food, 2 shelter, etc. Each camp has a state vector

$$s =< s_1, s_2, ..., s_k >$$

where $s_i = 1$ if the camp has need i and 0 otherwise. For simplicity we assume the need of the camp is given by

$$Ns =< Ns_1, Ns_2, ..., Ns_k >$$

where N is the number of people in the camp.

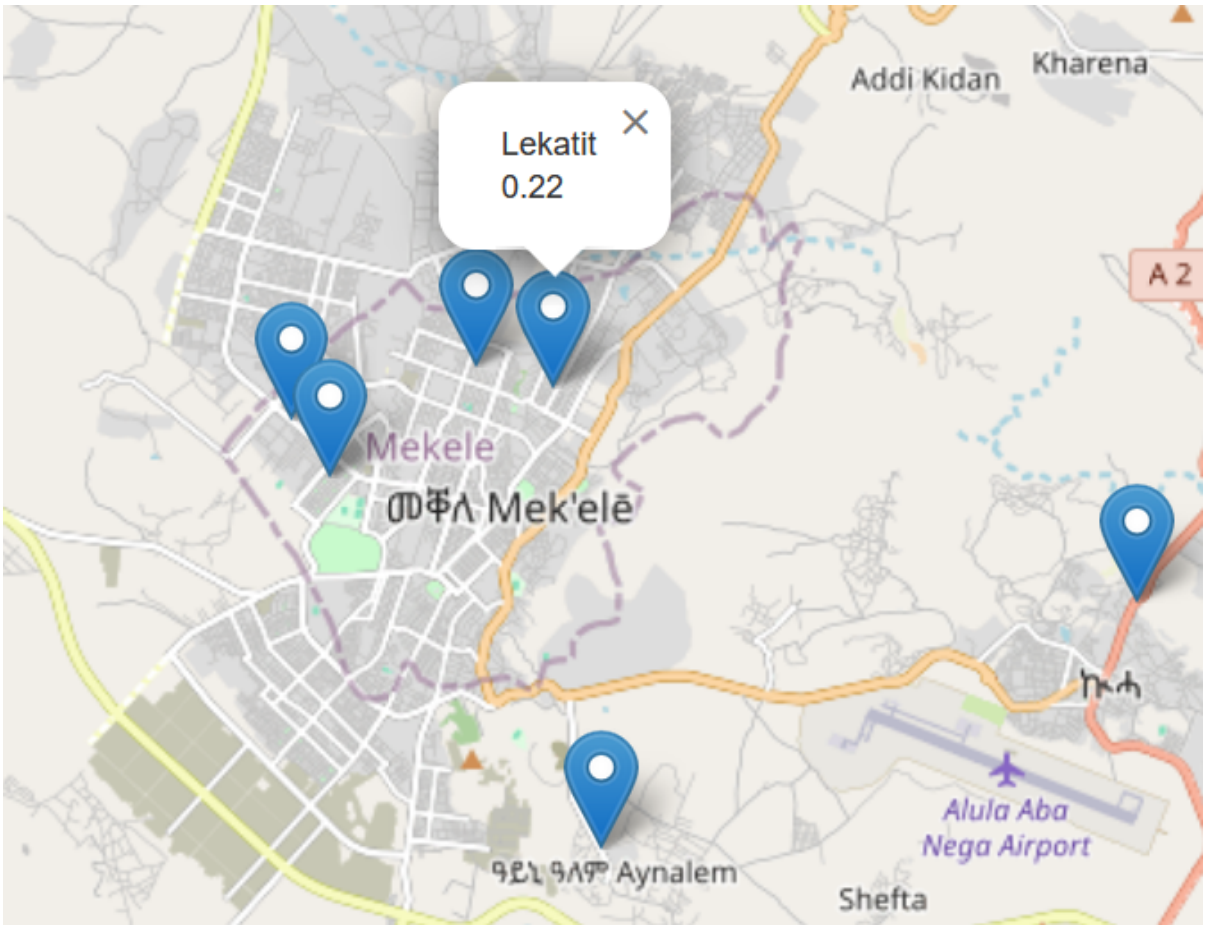
Each NGO has a capacity vector of the form

$$< n_1, n_2, ..., n_k >$$

where n_i indicates the number of people for which it can supply need i. An IDP camp with need Ns is feasible for an NGO if and only if

$$Ns =< Ns_1, Ns_2, ..., Ns_k > \preceq < n_1, n_2, ..., n_k >,$$

where $p \preceq q$ if each component of p is less than or equal to the corresponding component of q. The figure below shows a basic response model output which maps effective response options for a hypothetical NGO. Clicking on an icon gives the camp name and proportion of children under 3. (See the JNB "IDP Response" available at https://drive.google.com/drive/folders/1zqQB-hEPocxOVOjiI0Q32XDKVJoq6PI3?usp=sharing.)



Hypothetical NGO's map of effective response IDP camp locations with proportion of children under age 3. See the JNB 'IDP Response' available at https://drive.google.com/drive/folders/1zqQB-hEPocxOVOjiI0Q32XDKVJoq6PI3?usp=sharing

ENTROPY IN SOCIETAL APPLIATIONS

Entropy for societal applications can be introduced in an intuitive way using simple examples. We suggest that there is more than one way to interpret the meaning of entropy depending on the context. The computation of entropy is based on empirical histograms whose choice of bins determines the meaning of higher or lower entropy.

Example 1: ENTROPY AS A MEASURE OF DISAGREEMENT

Suppose a campus club's leadership committee comprised of N students is deliberating the use of funding for a certain year-end activity and must decide between Option A and Option B. The table below shows several possibilities where a proportion $p_1 = p$ of the committee strongly prefers option A and a proportion $p_2 = 1 - p$ strongly prefers option B. The degree of disagreement is reflected by the (Shannon) entropy defined as

$$H(p_1, p_2) = -\sum_{i=1}^{2} p_i \ln p_i = -[p \ln p + (1-p)\ln(1-p)],$$

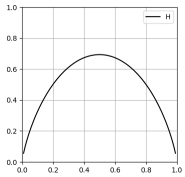
where $0 \leq p \leq 1$.

Committee Preferences	p1	p2	Disagreement	H
All favor Option A	1	0	none	0
All favor Option B	0	1	none	0
90% favor Option A	.9	.1	small	.325
50% favor each option	.5	.5	maximum	.69

Note

- H=0 when $p_1 = 1$,$p_2 = 0$ or when $p_1 = 0$,$p_2 = 1$.
- H has a maximum at $p_1 = p_2 = .5$ as can be seen from a graph of $H(p) = -[p \ln p + (1-p)\ln(1-p)]$ (see Figure below) and verified using calculus:

$$H'(p) = -\ln p - 1 + 1 + \ln(1-p) = 0 \Rightarrow \ln(\frac{1-p}{p}) = 0 \Rightarrow \frac{1-p}{p} = 1 \Rightarrow p = .5.$$



Note that the entropy generalizes to any number of options $n \geq 2$:

$$H(p_1, p_2, ..., p_n) = -\sum_{p_i > 0} p_i \ln p_i.$$

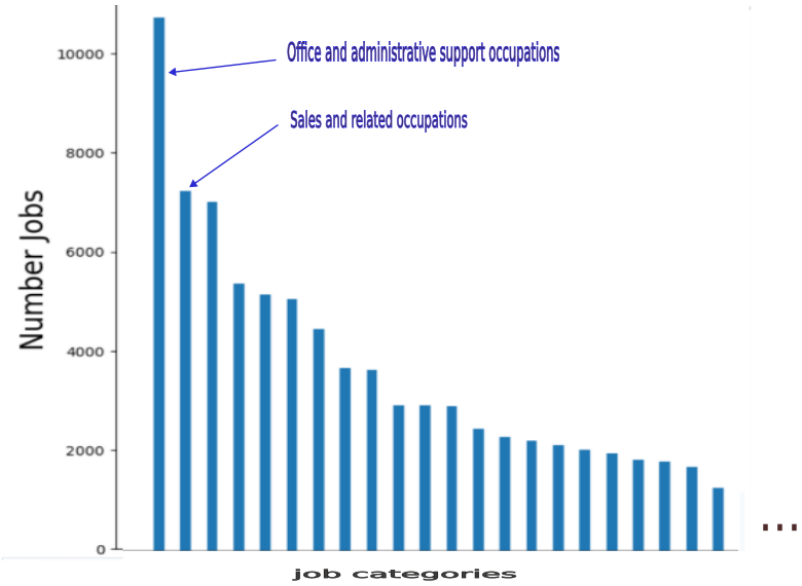
where $p_i \geq 0$ and $\sum p_i = 1$. For the case where the committee is split evenly between the n options ($p_i = 1/n$ for $i = 1, 2, ..., n$), the entropy is

$$H = -\sum_{i=1}^{n} (1/n)\ln(1/n) = \ln n.$$

Thus, for an even split between two options, the level of disagreement is $H = \ln 2 \approx .69$ and for three options it is $\ln 3 \approx 1.1$.

Example 2 ENTROPY AS A MEASURE OF DIVERSIFICATION

Business types have various classifications, including the North American Industry Classification System (NAICS), a hierarchical taxonomy with resolution levels 2-6. Level 2 describes broad sectors such as "71= arts and entertainment" and level 6 the most granular classifications of businesses such as "711110= dinner theater". Using data from the US Bureau of Labor Statistics https://www.bls.gov/oes/tables.htm, a histogram such as Figure 4 gives in order of decreasing frequency the number of jobs in each of N job categories.



Data for this histogram specifies empirical probabilities that a job is of a given category. The Shannon entropy is computed from the histogram data as $H = -\sum_{p_i>0} p_i \ln p_i$ where p_i is the proportion of jobs belonging to job type i and N is the number of job types ($N = 290$ and $H = 7.25$ for Abilene. (See the JNB "Job Diversification" available at https://tinyurl.com/3awd2kvx.)

This entropy can be used to compare the level of job diversification in different cities. A higher entropy reflects greater diversification of employment.

Example 3: ENTROPY AS A MEASURE OF TEST DEFICIENCY

1) Let's specify the distribution of number of test failures (0,1,2, or 3) for six sections of a calculus course. For example,

in section S1, 30 students have zero test failures.

In section S2, 19 students have zero test failures and 1 student has one test failure.

In section S3, 15 students have zero test failures, 5 students have one test failure, 3 students have two test failures, and 2 students have three test failures.

In section S4, 1 student has zero test failures, 5 students have two test failures, and 4 students have three test failures.

In section S5, the number of failures is exactly double that of section S4.

Section S6 resembles section S2, with 19 students in one bin and 1 student in another bin. (The numbers 19 and 1 are the same, but the deficiencies are completely different.)

```
S1=[30,0,0,0]
S2=[19,1,0,0]
S3=[15,5,3,2]
S4=[1,0,5,4]
S5=[2,0,10,8]
S6=[1,0,0,19]
```

Note that the total number N of students in all six sections is N=30+20+25+10+20+20=125 and the mean number of tests failed $U = 57/125 \approx .456$.

2) The mean number of tests failed in a section (m), the weighted mean (w_m) Shannon entropy (H) and weighted Shannon entropy for deficiency (H_d) are given respectively by

$$m = \sum_{p_i>0} i\, p_i = \sum_{n_i>0} i\, \frac{n_i}{n}$$

$$m_w = \frac{1}{N} \sum_{p_i>0} n_i\, i\, p_i = -\frac{1}{N} \sum_{n_i>0} n_i\, i\, \frac{n_i}{n}$$

$$H = -\sum_{p_i>0} p_i \ln p_i = \sum_{n_i>0} \frac{n_i}{n} \ln(\frac{n_i}{n}),$$

and

$$H_d = -\frac{1}{N} \sum_{p_i>0} n_i\, i\, p_i \ln(p_i) = -\frac{1}{N} \sum_{n_i>0} n_i\, i\, \frac{n_i}{n}\, \ln(\frac{n_i}{n}),$$

where

i) N is the total number of students in all sections

ii) i=number of deficiencies (tests failed)

iii) n_i =number of students in a given section failing i tests

iv) $p_i = n_i/n$ where n is the number of students in the given section

3) We can now compute the entropy H, deficiency entropy H_d , mean m, and weighted mean w_m for each section.

```
TestD(S1)
```

```
Test Failures [30, 0, 0, 0]
Entropy H= 0.0
mean m= 0.0
weighted mean m_w= 0.0
Deficiency Entropy Hd= 0.0
```

```
TestD(S2)
```

```
Test Failures [19, 1, 0, 0]
Entropy H= 0.199
mean m= 0.05
weighted mean m_w= 0.0
Deficiency Entropy Hd= 0.001
```

```
TestD(S3)
```

```
Test Failures [15, 5, 3, 2]
Entropy H= 1.085
mean m= 0.68
weighted mean m_w= 0.018
Deficiency Entropy Hd= 0.035
```

```
TestD(S4)
```

```
Test Failures [1, 0, 5, 4]
Entropy H= 0.943
mean m= 2.2
weighted mean m_w= 0.078
Deficiency Entropy Hd= 0.063
```

```
TestD(S5)
```

```
Test Failures [2, 0, 10, 8]
Entropy H= 0.943
mean m= 2.2
weighted mean m_w= 0.157
Deficiency Entropy Hd= 0.126
```

```
TestD(S6)
```

```
Test Failures [1, 0, 0, 19]
Entropy H= 0.199
mean m= 2.85
weighted mean m_w= 0.433
Deficiency Entropy Hd= 0.022
```

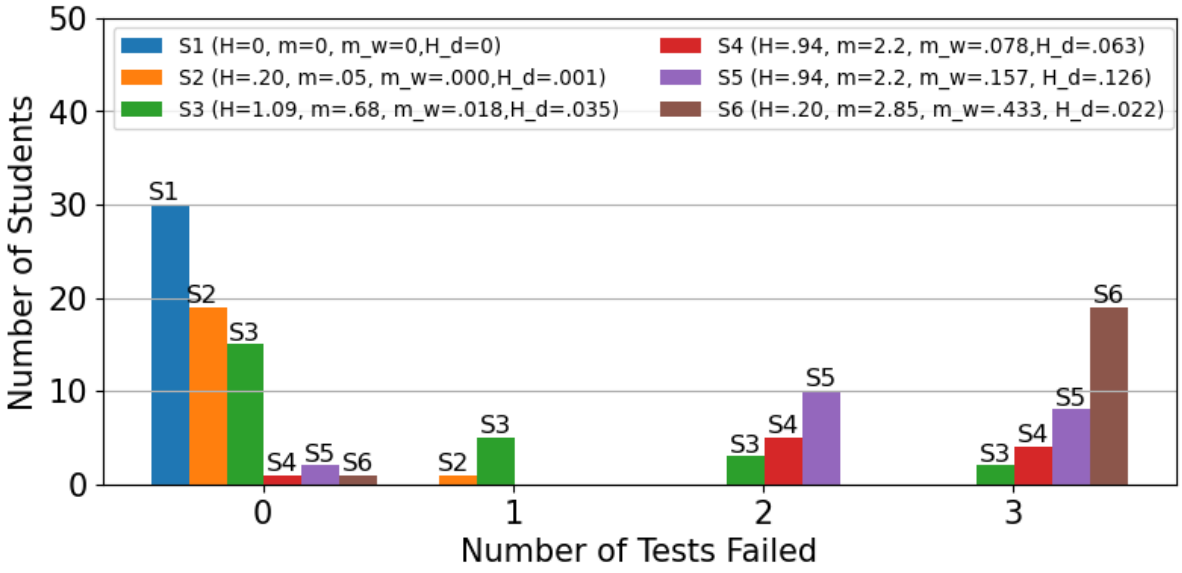
4) We can create a histogram of the data

```
import matplotlib.pyplot as plt

plt.figure(figsize=(9.15, 4))

count=('0','1','2','3')
frequency = {
    'S1 (H=0, m=0, m_w=0,H_d=0)': S1,
    'S2 (H=.20, m=.05, m_w=.000,H_d=.001)': S2,
    'S3 (H=1.09, m=.68, m_w=.018,H_d=.035)': S3,
    'S4 (H=.94, m=2.2, m_w=.078,H_d=.063)': S4,
    'S5 (H=.94, m=2.2, m_w=.157, H_d=.126)':S5,
    'S6 (H=.20, m=2.85, m_w=.433, H_d=.022)':S6
}
x = np.arange(len(count))  # the label locations
width=.15
multiplier=.2

for attribute, measurement in frequency.items():
    offset = .125+width * multiplier
    rects = plt.gca().bar(x +offset, measurement, width, label=attribute)
    #plt.gca().bar_label(rects, padding=3)
    multiplier += 1

for i in np.arange(0,4,1):
    if S1[i]>0:
        plt.gca().text(x[i]+.07,S1[i],'S1',size=12,ha='left',va='bottom')
    if S2[i]>0:
        plt.gca().text(x[i]+.22,S2[i],'S2',size=12,ha='left',va='bottom')
    if S3[i]>0:
        plt.gca().text(x[i]+.39,S3[i],'S3',size=12,ha='left',va='bottom')
    if S4[i]>0:
        plt.gca().text(x[i]+.54,S4[i],'S4',size=12,ha='left',va='bottom')
    if S5[i]>0:
        plt.gca().text(x[i]+.69,S5[i],'S5',size=12,ha='left',va='bottom')
    if S6[i]>0:
        plt.gca().text(x[i]+.84,S6[i],'S6',size=12,ha='left',va='bottom')
# Add some text for labels, title and custom x-axis tick labels, etc.
plt.gca().set_xlabel('Number of Tests Failed',size='15')
```

(continues on next page)

(continued from previous page)

```
plt.gca().set_ylabel('Number of Students',size='15')
plt.gca().set_xticks(x + 3.5*width, count)
plt.gca().legend(loc='upper left', ncols=2,fontsize='9.75')
plt.gca().set_ylim(0, 50)
plt.xticks(fontsize= 15)
plt.yticks(fontsize= 15)
plt.grid(axis='y')
plt.savefig("fig1.png")
plt.show()
```



Exercises

4.4.1

Suppose a group of x NGO's denoted $NGO_1, ..., NGO_x$ are considering a response in a region R with y camps denoted $Camp_1, ..., Camp_y$ (we assume $x \leq y$). An effective total response is of the form

$$< a_1, a_2, ..., a_x >$$

where $Camp_{a_i}$ is feasible for NGO_i $(i = 1, 2, ..., x)$. What is the probability of a random assignment being effective if there are f effective total responses?

4.4.2 Suppose NGO_i has a utility function which measures its preference for a feasible assignment to a camp a_i. Utility is measured by efficiency. For example, a camp which is easier to reach (lower total cost to transport staff and materials) would have higher utility. Explain how effective total responses might be rank-ordered by efficiency.

4.4.3 How might a basic response model be modified in the case where there are no effective total responses (overwhelming needs) ?

References

Amdat, W.C. (2022). "The Chicago Hardship Index: An Introduction to Urban Inequity." Available at https://www.tandfonline.com/doi/full/10.1080/26939169.2021.1994489

Bertin, E. (2021). Statistical Physics of Complex Systems: A Concise Introduction (3rd Edition). Cham, Switzerland: Springer Nature.

GAISE College Report ASA Revision Committee, (2016), "Guidelines for Assessment and Instruction in Statistics Education College Report 2016," available at: https://www.amstat.org/asa/files/pdfs/GAISE/GaiseCollege_Full.pdf

Hartnett, K. (2022), "How Shannon entropy imposes fundamental limits on communication," available at https://www.quantamagazine.org/how-claude-shannons-concept-of-entropy-quantifies-information-20220906/

Lesser, L. (2007),"Critical Values and Transforming Data: Teaching Statistics with Social Justice." Journal of Statistics Education, 15:1, DOI:10.1080/10691898.2007.11889454

Nawrock, D.N. and Harding W.H. (1986), "State-value weighted entropy as a measure of investment risk," available at https://www.researchgate.net/publication/24072353_State-Value_Weighted_Entropy_as_a_Measure_of_Investment_Risk

Saraiva, P. (2023) "On Shannon entropy and its applications." Available at https://www.sciencedirect.com/science/article/pii/S2307410823000433

Schramm, H., Koehlmoos T., and Nestler, S. (2018), "Statistics in Pursuit of Social Justice," available at:https://csbaonline.org/about/news/statistics-in-pursuit-of-social-justice

Wang, L., Whittemore, K., Johnston, S.A., and Stafford P. (2017), "Entropy is a simple measure of the antibody profile and is an indicator of health status: proof of concept," available at https://www.nature.com/articles/s41598-017-18469-6}

15.5 Solution to Exercises

15.5.1 Section 2

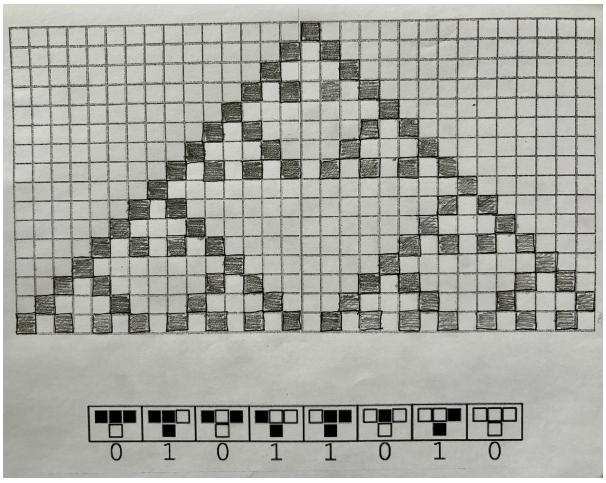
2.1 A puzzle with only 1 or 2 types of pieces will be less complex than a jigsaw puzzle where each piece is unique. For complex problems in human society, a major challenge is how to limit the characteristics of constituent parts so the entire population is subdivided into major types whose analysis gives a good picture of the state of the system as a whole. (Such a strategy is also used within statistical physics in the form of the re-normalization group approach [Bertin 2021].)

2.2.1 Rebuilding may be impossible within an isolated rural community. In a major metropolitan area such as Mekelle, the regional capital, distance learning facilitated by international money transfer can turn an isolated learning system into a closed system (transfer of 'energy' in the form of credit and knowledge but not physical resources).

The post-war rural Embasneyti school district became an open system in August 2023 when it received a 'mass transfer' of 80 blackboards, 6000 notebooks, pens and pencils, 3 laptops and 16 soccer balls. This enabled the relaunch of school for 6,000 children in the district, and was a story covered by regional TV media (https://www.facebook.com/yohans.teklemariam/videos/1018050329637169/?d=w&mibextid=2PtUEq)

2.2.2 If each IDP camp had the same characteristics (size, types of needs, registration with the government etc.) response to the IDP crisis would be much simplified due to coherence. In the absence of such coherence, a survey of IDP camps may be administered on an ongoing basis based on criteria such as shown in **Figure 20**. The response data could then be used to measure similarity (correlation) between camps as well as track changes to each camp over time.

2.2.3 When a sheet of graph paper is filled using Wolfram's Rule 90, the result is shown in the figure below. Note the emergence of a fractal structure (Sierpinski triangle).



f 2.3.1 Let λ_1, λ_2 be the two eigenvalues of the Jacobian matrix, A. The qualitative behavior of the nonlinear system near P depends on the signs of the eigenvalues. When both eigenvalues are real: If $\lambda_1 > \lambda_2 > 0$, the system will be an unstable node, shown in the far right of quadrant I in **Figure 7**. If $\lambda_1 < \lambda_2 < 0$, the system will be a stable node, as shown in the far left of quadrant II in **Figure 7**. When $\lambda_2 < 0 < \lambda_1$, the system will behave as a saddle point, as is shown in **Figure 7**. in quadrants III and IV.

The cases when there are complex eigenvalues are shown in the region above the parabola $4detJ = tr(J)^2$. In this case, the eigenvalues are $\lambda_{1,2} = \alpha \pm i\beta$. Then, when $\alpha < 0$, the system will be a stable spiral. When $\alpha > 0$, the system will be an unstable spiral. Finally, when $\alpha = 0$, the system will be a centre, shown along the y-axis of **Figure 7**.

2.3.2 Let J denote the Jacobian matrix $\begin{pmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{pmatrix}$.

In our case, $\frac{dx}{dt} = 3x - y^2 = f(x,y)$ and $\frac{dy}{dt} = sin(y) - x = g(x,y)$, so we have $J = \begin{pmatrix} 3 & -2y \\ -1 & \cos(y) \end{pmatrix}$. At $(0,0)$, $J = \begin{pmatrix} 3 & 0 \\ -1 & 1 \end{pmatrix}$. The eigenvalues of $J(0,0)$ are $\lambda = 1, 3$. Therefore, this system is a source, and the equilibrium (0,0) is unstable.

2.4.1 See the JNB Lab "Complex Systems" available at the end of this chapter.

2.4.2 As can be shown in Exercise 2.4.1, when $K > 0$ is small, a stable steady state occurs at $(0, C_*)$ with $C_* > 0$ For higher values of K, the equilibrium point $(0, C_*)$ loses its stability and there is a new stable steady state (F^*, C^*) with $F^* > 0, C^* > 0$. For even higher values of K, the equilibrium (F^*, C^*) is a spiral sink. For still higher values of K, the spiral equilibrium (F^*, C^*) loses its stability and a stable limit cycle (closed curve) solution appears. The bifurcation diagram follows this pattern, showing a stable equilibrium ($F_* = 0, C_* > 0$) for $K < k_1$. This equilibrium loses its stability at $K = k_1$ and there arises a new stable equilibrium (F^*, C^*) with $F^* > 0$ and $C^* > 0$. At $K = k_2$, a Hopf bifurcation occurs: the stable equilibrium point (F^*, C^*) loses its stability and a stable limit cycle appears for $K > k_2$.

2.4.3 a) To find a period one point for $x_{n+1} = rx_n(1 - x_n)$, we substitute x_n for x_{n+1} and solve for x_n.

$$x_n = rx_n(1 - x_n)$$

$$0 = -rx_n + rx_n^2 + x_n$$

$$0 = x_n(1 - r) + rx_n^2$$

$$0 = x_n(1 - r + rx_n)$$

So, $x_n = 0$, $x_n = \frac{r-1}{r}$ are the two fixed points we find. Therefore, when $1 < r < 2$, $\frac{r-1}{r}$ is a period one point for this map.

b) The Feigenbaum constant $\delta = \lim_{n\to\infty} \frac{r_{n-1}-r_{n-2}}{r_n - r_{n-1}} = 4.669\,201\,609\,...$ is a universal constant for period doubling bifurcations in 1D maps.

2.5 See the JNB Lab "Complex Systems" at the end of this chapter.

15.5.2 Section 3

3.1.1 We know that $\frac{dx}{dt} = v$ and $v = \frac{p}{m}$. So, $\frac{dx}{dt} = \frac{\partial H}{\partial p}$ since the partial derivative of H with respect to p is $\frac{p}{m}$.

We also know that $\frac{dp}{dt} = F$ and $F = -kx$. So, $\frac{dp}{dt} = \frac{-\partial H}{\partial x}$ since the partial derivative of H with respect to x is kx.

3.1.2 $\frac{dE}{dt} = (\frac{\partial H}{\partial x}\frac{dx}{dt}) + (\frac{\partial H}{\partial p}\frac{dp}{dt}) = (\frac{-dp}{dt}\frac{dx}{dt}) + (\frac{dx}{dt}\frac{dp}{dt}) = 0.$

3.1.3 Let $H = H(x^*, p^*) = \frac{(p^*)^2}{2m} + \frac{1}{2}k(x^*)^2$.

$\frac{dx^*}{dt^*} = -\frac{dx(t^*)}{dt} = v^* = \frac{p^*}{m} = \frac{dH}{dp^*}$.

$\frac{dp^*}{dt^*} = -m\frac{dv(t^*)}{dt^*} = m\frac{dv(t^*)}{dt} = ma(t^*) = -kx(t^*) = -kx^* = -\frac{\partial H}{\partial x^*}$.

3.2.1 First, observe that $E = -h\sum_{i=1}^{N} s_i = -hM, \qquad (s_i = \pm 1,)$

Next, notice that:

$$M = N_+ - N_= N_+ - (N - N_+) = 2N_+ - N.$$

Similarly,

$$M = -2N_- + N.$$

Next, we substitute in the values we just solved for M. We will first look at N_+

$$E = -h(2N_+ - N) = -2hN_+ + hN \Rightarrow 2hN_+ = hN - E \Rightarrow$$

$$N_+ = \frac{1}{2}\left(N - \frac{E}{h}\right)$$

Similarly for N_-, we substitute in the value we just solved for M. This leads to $E = -h(-2N_- + N)$ and by algebra: $N_- = \frac{1}{2}\left(\frac{E}{h} + N\right)$.

3.2.2 First, we know that

$\ln(N!) \approx N\ln(N) - N$. So,

$\mathbf{S}(E) = \ln\Omega(E) = \ln(\frac{N!}{N_+!N_-!}) = \ln(N!) - \ln(N_+!) - \ln(N_-!) \approx N\ln N - N - (N_+\ln N_+ - N_+) - (N_-\ln N_- - N) = (-N + N$

So, $\mathbf{S}(E) \approx \ln\frac{N^N}{N_-^{N_-}N_+^{N_+}}$, as desired.

3.2.3

$$\Omega_{tot}(E_S|E_{tot}) = \Omega_S(E_S)\Omega_S(E_{tot} - E_S)$$

By maximum-likelihood, $\frac{\partial}{\partial E_S}\ln\Omega_S(E_S) + \frac{\partial}{\partial E_S}\ln\Omega_S(E_{tot} - E_S) = 0$

$$\Rightarrow \frac{\partial}{\partial E_S}\ln\Omega_S(E_S) + \frac{\partial}{\partial E_R}\ln\Omega_S(E_R)\frac{\partial E_R}{\partial E_S} = 0$$

$$\Rightarrow \frac{\partial}{\partial E_S}\ln\Omega_S(E_S) - \frac{\partial}{\partial E_S}\ln\Omega_R(E_R) = 0$$

$$\frac{\partial}{\partial E_S}\ln\Omega_S(E_S) = \frac{\partial}{\partial E_S}\ln\Omega_R(E_R)$$
$$\frac{\partial \ln\Omega_S}{\partial E_S} = \frac{\partial \ln\Omega_R}{\partial E_R}$$

3.2.4 The formula for the grand-canonical distribution is merely a specific term divided by all the terms in the distribution. First, we must assume the system exchanges both energy and particles with the environment. Intuitively, this formula is finding the probability by dividing the particular configuration by the grand-canonical partition function, which is the sum of all possible configurations. So, this gives the probability that the particular configuration (C_S) occurs.

3.2.5 a)

$$-\sum_C P(C)\ln P(C) = -\sum_C \frac{1}{\Omega(E)}\ln\frac{1}{\Omega(E)}$$
$$= -\frac{1}{\Omega(E)}\sum_C \ln(\frac{1}{\Omega(E)})$$
$$= -\frac{1}{\Omega(E)}\Omega(E)\ln\left(\frac{1}{\Omega(E)}\right)\ln\Omega(E) = \mathbf{S}.$$

b)

$$\mathbf{S} = -\sum_C \frac{1}{Z}e^{-E(C_S)/T}\ln\left(\frac{1}{Z}e^{-E(C_S)/T}\right)$$
$$= -\sum_C \frac{1}{Z}e^{-E(C_S)/T}\left(\ln\left(\frac{1}{Z}\right) - E(C_S)\beta\right)$$
$$= [\frac{\ln Z}{Z}\sum_C e^{-E(C_S)/T}] + < E(C_S) > \beta$$
$$= \ln Z + \beta < E(C_S) >$$

3.3.1 Let $m = \frac{1}{N}\sum_{i=1}^N s_i$. Then

$$-\frac{J}{2}Nm^2 = -\frac{J}{2N}(\sum_{i=1}^N s_i)^2 = -\frac{J}{2N}(2\sum_{i<j} s_i s_j + \sum_{i=1}^N s_i^2) = -\frac{J}{N}\sum_{i<j} s_i s_j - \frac{J}{2N}N = -\frac{J}{N}\sum_{i<j} s_i s_j + E_0 = E_R$$

where $E_0 = -\frac{J}{2}$.

3.3.2 Taking the derivative of the function $f_T(m)$ and setting it equal to zero, we obtain

$$f_T'(m) = (1 - \frac{1}{T})m + \frac{1}{3}m^3 = m[1 - \frac{1}{T} + \frac{1}{3}m^2] = 0.$$

The roots are

$$m = 0$$

and

$$m = \pm\sqrt{\frac{3-3T}{T}}.$$

If $T \geq 1 = T_{crit}$, then $m = 0$ is the only real root and is a minimum since $f_T''(m) = 1 - \frac{1}{T} + m^2$ is positive when $m = 0$ and $T > 1$.

If $T < 1$, then a minimum occurs at $\pm m_0$ where $m_0 = \sqrt{\frac{3}{T} - 3}$. (Note that $f_T''(m_0) = \frac{2}{T} - 2 > 0$.)

15.5.3 Section 4

4.1 Similarly to the steady state for 1980, for the steady state for 2005, x_1, x_2, x_3 as steady state values must by definition satisfy $x_1 + x_2 + x_3 = 1$ and also

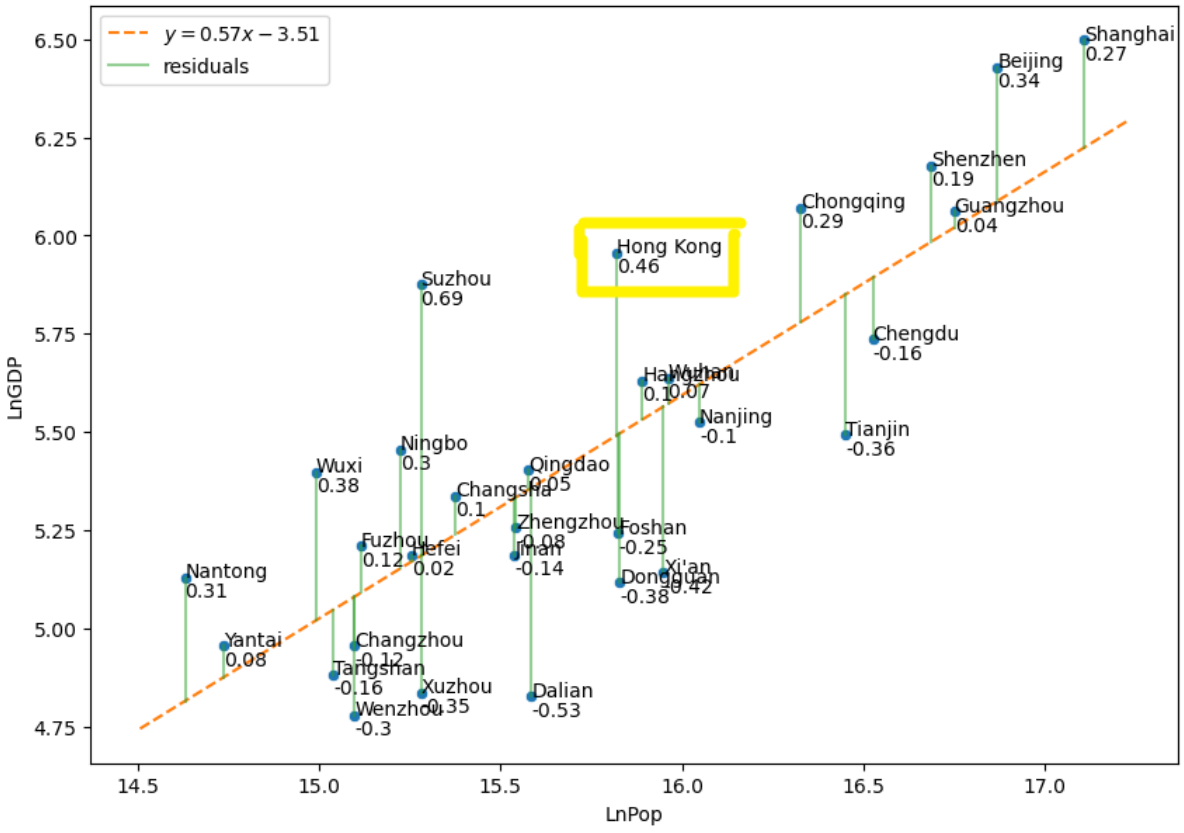
$$\begin{pmatrix} .998 & .127 & .297 \\ .002 & .506 & .293 \\ 0 & .367 & .409 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

Once again, the steady state is an eigenvector of the transition matrix M with eigenvalue 1:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} .9897 \\ .0063 \\ .0039 \end{pmatrix}$$

Thus, the steady state for the Not Incarcerated, Not on Parole population decreased slightly while the steady states for the Incarcerated population and Parole population both increased slightly from 1980 to 2005.

4.2 If Hong Kong is added into the SAMI graph, the updated SAMI graph should look like the one below, where Hong Kong is highlighted:



See the JNB Lab "Complex Systems" at the end of this chapter.

4.3

a)

Class	Number Students	Number Late Assignments			
		0	1	2	3
C1	30	30	0	0	0
C2	20	19	1	0	0
C3	25	15	5	2	3

b) Class C1 (all students turned in all assignments on time) is the most orderly and has entropy $H = -1\ln(1) = 0$. Class C2 was very orderly (only 1 out of 20 students had 1 late assignment) and a low entropy value ($H = -[.95\ln(.95) + .05\ln(.05)] = .20$). Class C3 had the highest entropy ($H = -[.6\ln .6 + .2\ln .2 + .09\ln .09 + .12\ln .12] = 1.1$) with a considerable proportion of students students having 1, 2, or 3 late assignments. This type of disorder can add a burden to the grading process and is one reason why teachers penalize or do not accept late assignments.

4.4.1 Note that there are $C(y,x) = \frac{y!}{x!(y-x)!}$ possible NGO assignments. Suppose f of these assignments are effective total responses. The probability of a random assignment being effective is $p = f/C(y,x)$.

4.4.2 The utility of an effective total response is the mean of the IDP assignment utilities for that response. Ordering assignments based on maximizing mean utility would thus be an efficient rank-ordering.

4.4.3 In cases where there are no effective total responses, a "need gap" might be introduced to measure how close an NGO is to meeting the need of a given IDP camp. Assignments could be made by minimizing the need gap for each NGO beginning with the NGO with the smallest amount of resources and ending with the NGO with the greatest amount of resources. The total need gap would be a measure of the total response effectiveness. A criteria such as maximizing child protection might be used to settle ties in total response effectiveness.

15.6 JNB LAB: Complex Systems

15.6.1 Bifurcations in the Rozenzweig-MacArthur Predator Prey (RMPP) Equations

In this section will look at Rozenzweig-MacArthur Predator-Prey (RMPP) Equations

(For background, see https://staff.fnwi.uva.nl/a.m.deroos/projects/QuantitativeBiology/43-HopfPoint-Rosenzweig.html)

Defining the equations RMPP is a 2D nonlinear system of ODEs

$$\frac{dF}{dt} = rF(1 - \frac{F}{K}) - \frac{aF}{1+ahF}C$$

$$\frac{dC}{dt} = \epsilon\frac{aF}{1+afF}C - \mu C$$

where:

$F =$ prey density (eg rabbits)

$C =$ predator density (eg foxes)

$\frac{dF}{dt}$ = dynamic change of prey

$\frac{dC}{dt}$ = dynamic change of predator

r = prey natural growth rate

K = prey carrying capacity

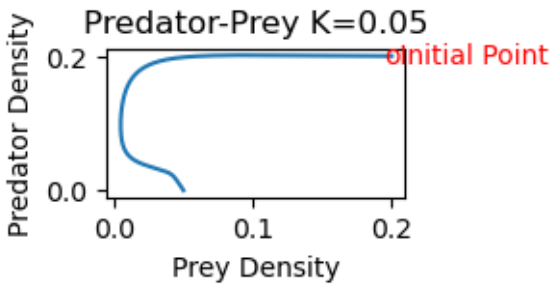
a = attack rate

h = handling rate (predator needs h time units to consume prey)

e = prey offspring proportionality constant to F

μ - predator death rate

The following cell simulates the solution to RMPP over the time interval $0 \le t \le 500$ with parameter values $r = .5, K = .05, a = 5, h = 3, e = .5, \mu = .1$



Note that the predators grow extinct and the prey density approaches a value of approximately .05.

References: https://staff.fnwi.uva.nl/a.m.deroos/projects/QuantitativeBiology/43-HopfPoint-Rosenzweig.html

Anthony Hills, https://github.com/INASIC/predator-prey_systems/blob/master/Modelling Predator-Prey Systems in Python.ipynb

Exercises

Exercises

1 a) Define a function rmpp(x0,y0,K) which whose inputs specify the initial point (x0,y0) and prey carrying capacity K and whose output is a graph of the solution to the RMPP equations for $0 \leq t \leq 500$.

b) Check that your output for rmpp(.2,.2,.05) agrees with the above plot.

2. a) Make a chart showing the solution behavior as K is increased to .15, .25 and .3. (Note: for k=.3, also check the solution behavior for (x0,y0)=(.1,.15)) b) Describe the difference in the solution behavior for these K values.

3) Explain why RMPP has a Hopf bifurcation (one in which there is a critical value at which a stable equilibrium becomes unstable and a stable periodic solution is created.)

15.6.2 Sensitive Dependence on Initial Conditions in a Fitzhugh Nagumo System

Imagine you are a new grad student. You are interested in applied math so are taking a course in ordinary differential equations (ODEs). One day, your professor talks about his colleague Dr. John Rinzel, who at the time was head of the math research branch at NIH (National Institutes of Health). Dr. Rinzel is an expert in mathematical neurobiology, and has developed several models of how neurons fire (active phase) and then relax (interval of quiescence). One of the nonlinear models he investigated is a 3 dimensional ODE system which we will refer to as the Fizhugh-Nagumo system (FHN):

$$\frac{dv}{dt} = v - v^3/3 - w + y + I$$

$$\frac{dw}{dt} = \phi(v + a - bw)$$

$$\frac{dy}{dt} = \theta(-v + c - dy)$$

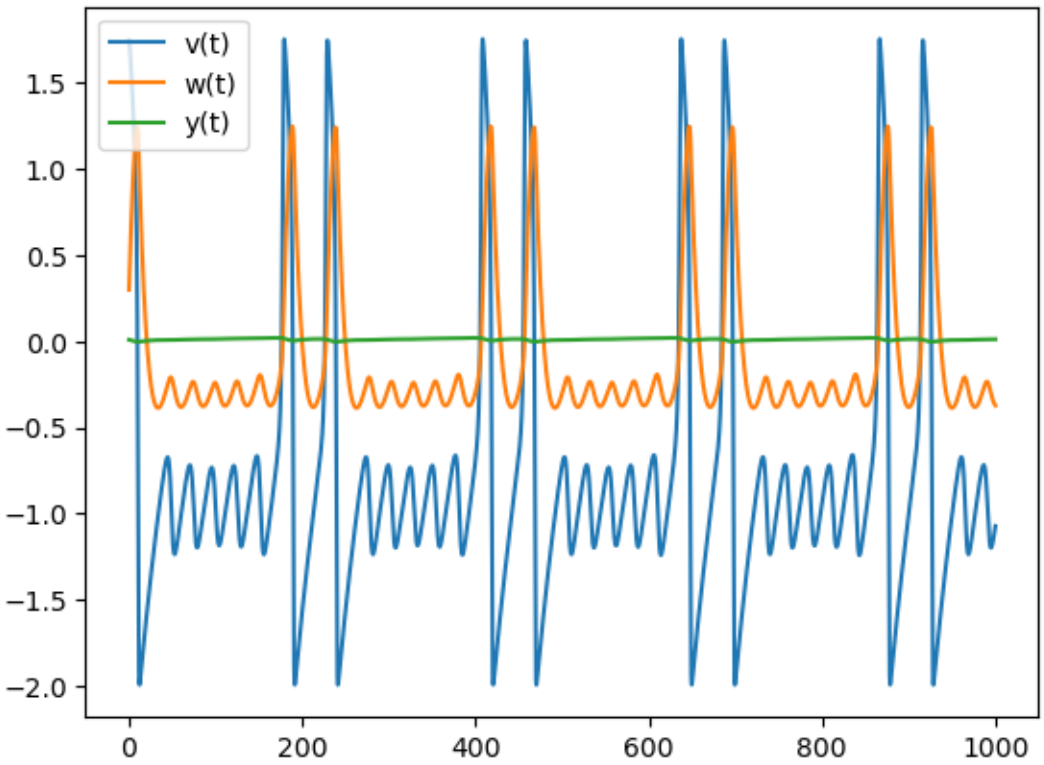
$v(t)$ is a membrane voltage variable which spikes when the neuron fires. $w(t)$ represents the activity of several membrane channel proteins which help facilitate the neuron's firing. The parameter I is a bifurcation parameter corresponding to the magnitude of the stimulus current. The underlying 2D system (without the variable y) undergoes a Hopf bifurcation. For $I < I_{crit}$, there is a stable spiral equilibrium point which corresponds to the interval of quiescence. For $I < I_{crit}$, the

equilibrium point loses its stability as there is a stable periodic solution which corresponds to the spiking of v during the active phase.

Note that in the 3D system, if I is fixed, the inclusion of y(t) will effectively move the 2D system back and forth across the bifurcation point of the 2D system.

Using parameter values and initial point specified by Dr. Rinzel, we solve the FHN system numerically and plot the solution to to see the action potentials.

```
[t,x_t]=solve_FHN([1.744,.298,.00872]) # Initial conditions v=1, w=0, y=0.
plt.plot(t,x_t)
plt.legend(["v(t)","w(t)","y(t)"])
plt.savefig("FHN1.png")
```



The periodic oscillation for v(t) has a stable burst pattern denoted 1100000 meaning that two large spikes (11) followed by five small oscilations (ooooo); this pattern will recur over and over.

Exercise

Exercise

1. Explain how the bursting behavior in FHN is related to a Hopf bifurcation in the underlying 2D system.
2. Nonlinear ODE systems can exhibit complex behavior such as sensitive dependence on initial conditions.

a) Re-simulate FHN using the same paramter values and the initial condition x0=[1.725,.337,.00764]. What do you observe? b) Why does FHN exhibit sensitive dependence on initial conditions and how does FHN exhibit this?

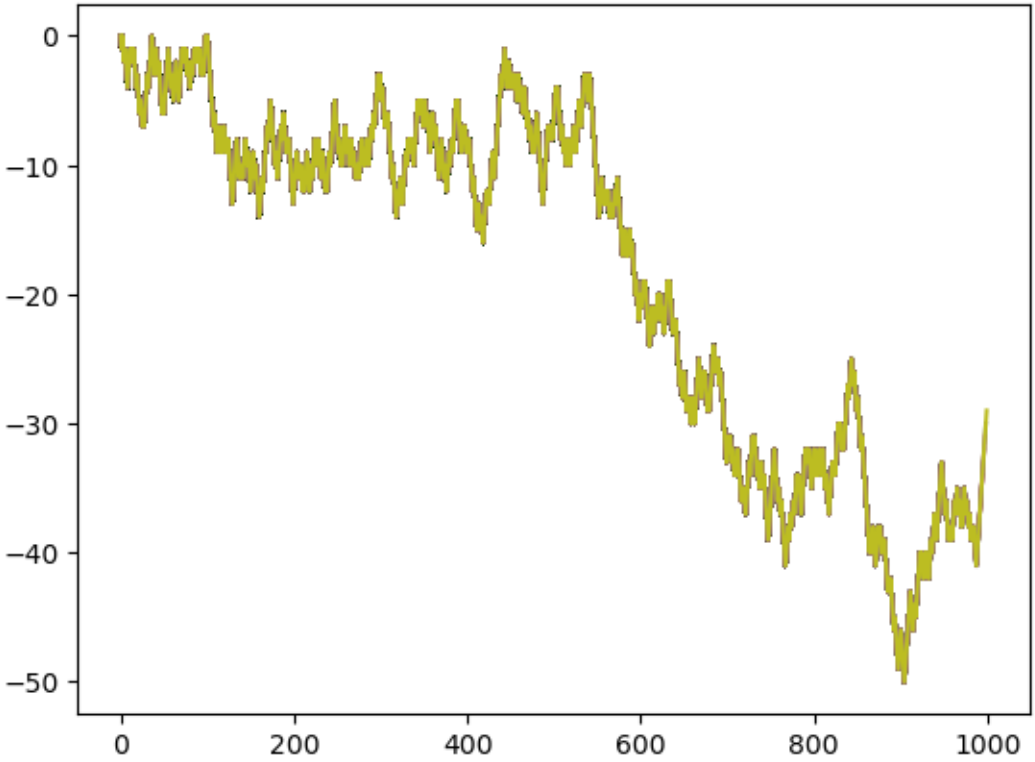
15.6.3 Random Walks

A random walk beginning at $x_0 = 0$, is such that x_t, the position at timestep $t = 0, 1, 2, ...$ is

$$x_{t+1} = x_t + e_t$$

where $e_t = \pm 1$ with equal probability. The random variables e_t and $e_{t'}$ $(t \neq t')$ are assumed to be independent and identically distributed (IID).

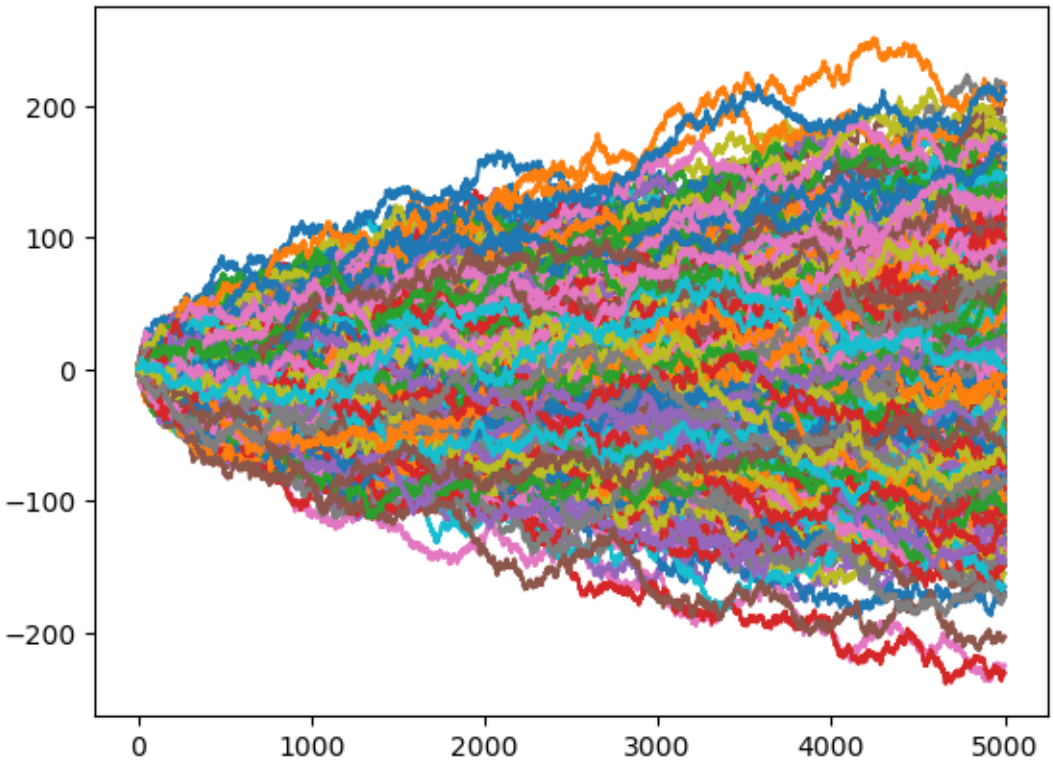
The following cell simulates a random walk.



Although each individual random walk is completely unpredictable, a large ensemble of random walks has a predictable structure since

$$< x_t >= 0$$

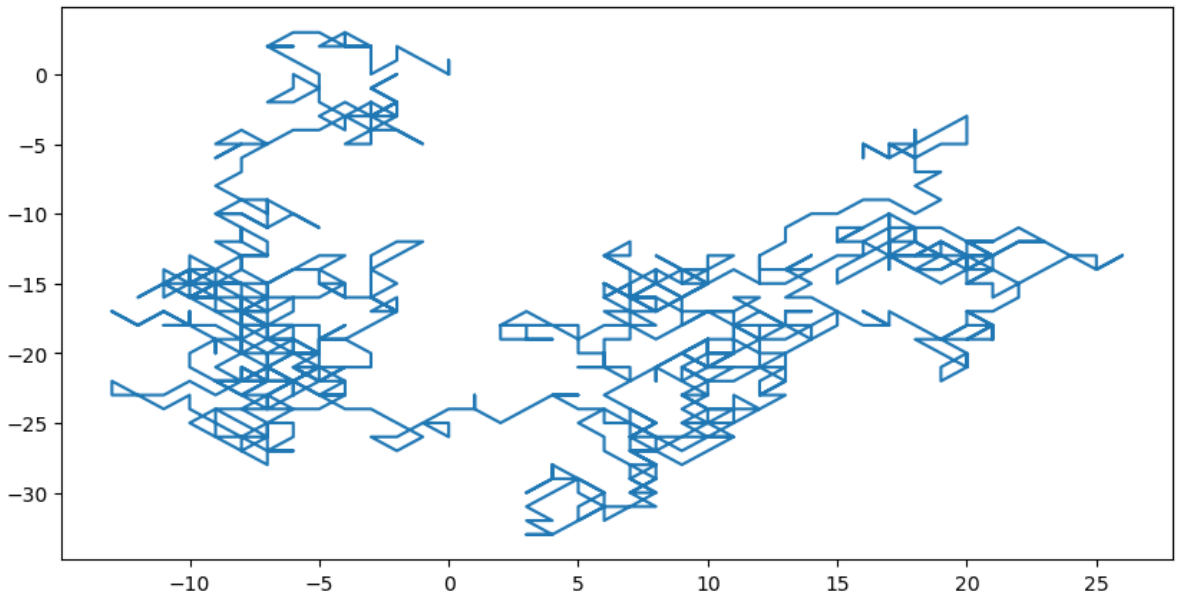
$$< x_t^2 >= t$$



Note that after 5000 steps, the mean for 1000 random walks of x_t (t=4999) is close to 0, and the mean of x_t^2 is close to t. In other words, for a normal random walk with $x_t = 0$, the mean is zero ($< x_t >= 0$) and the variance grows linearly with t ($< x_t^2 >= t$):

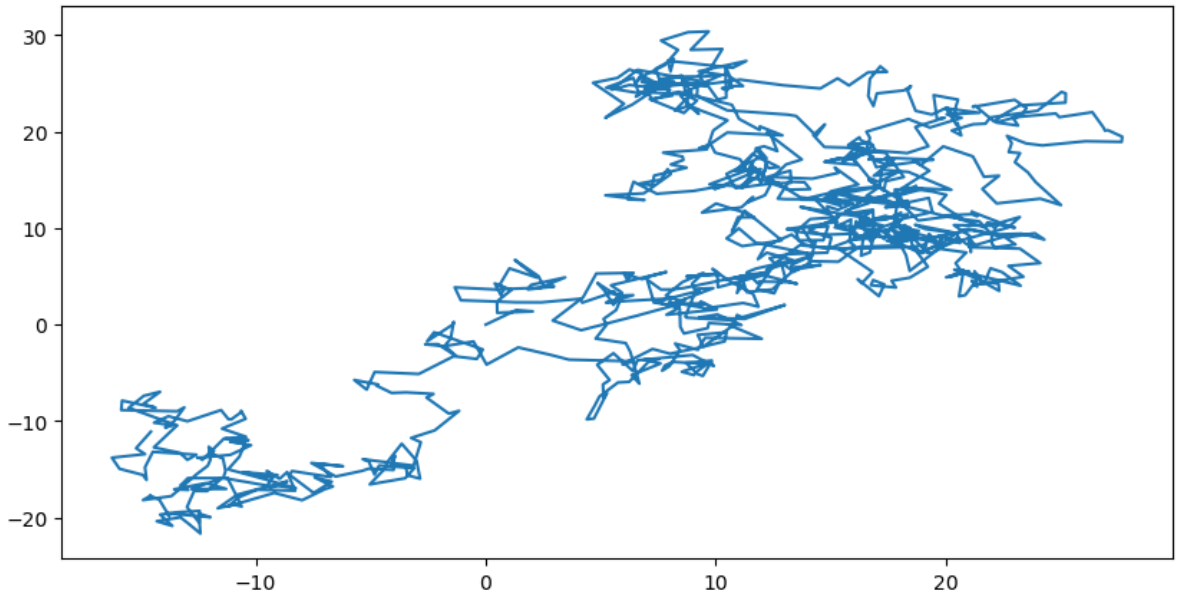
```
Mean of x[4999]= 0.566
Mean of x[4999]^2= 5195.856
```

Here is a graph of a 2D random walk with 1000 steps (both the x position and y position are random walks.)



Brownian Motion

We can also create a random walk using variable step sizes by drawing from a standard normal distribution (np.random.normal(0, 1, 1000))



Exercise

Exercise

1. a) A standard Cauchy distribution with p.d.f. $f(x) = \frac{1}{\pi(1+x^2)}$ is a heavy tail distribution with $< x >= 0$ and infinite variance. Simulate a 2D random walk with $N = 1000$ steps where each step is determined by a random draw from a Cauchy distribution (np.random.standard_cauchy(N)).

b) What difference do you notice about this random walk compared with brownian motion.

2. a) Plot the probability density function of a standard Cauchy distribution and a standard normal distribution. Explain why the former is said to have a heavy tail.

b) What is the significance of 'long flights' in Cauchy random walks?

15.6.4 Scale Adjusted Metropolitan Index (SAMI)

Urban scaling theory conceptualizes cities as interdependent socioeconomic and spatial (infrastructural) networks that coevolve to support each other. Major goals include:

- Find empirical characteristics of cities as a whole (extensive properties) such as GDP or road surface area which exhibit scale invariant relations (vary on average with city size).
- develop an urban scaling theory to explain the observed patterns
- develop predictive models

An extensive property of cities has the form

$$Y(N_i(t), t) = Y_i(t) = Y_0(t)N_i(t)^{\beta}e^{\xi_i(t)} > 0$$

- $i = 1, ..., N_c$ correspond to cities
- N_i is a measure of system size varies continuously with time t
- $Y_0(t)$ is a prefactor independent of scale
- β is a scaling exponent (elasticity) expressing the average relative percentage change in Y given a percentage variation in N at a fixed time t.

$$\beta = \frac{\frac{dY}{Y}}{\frac{dN}{N}}$$

- $\xi_i(t)$ are residuals which account for deviations of city i from the average scaling pattern of all cities. The residuals satisfy

$$\sum_{i=1}^{N_c} \xi_i = 0.$$

- Scaling laws arise when the variance $\sigma^2 = \frac{1}{N_c} \sum_{i=1}^{N_c} \xi_i^2$ is scale independent.

Scaling laws

A socio-economic variable (Y) follows a population (N) scaling law if

$$Y(N) = GN^{\beta}$$

with $G > 0$ and β are constants independent of the value of N.

Example

GDP of cities in China.

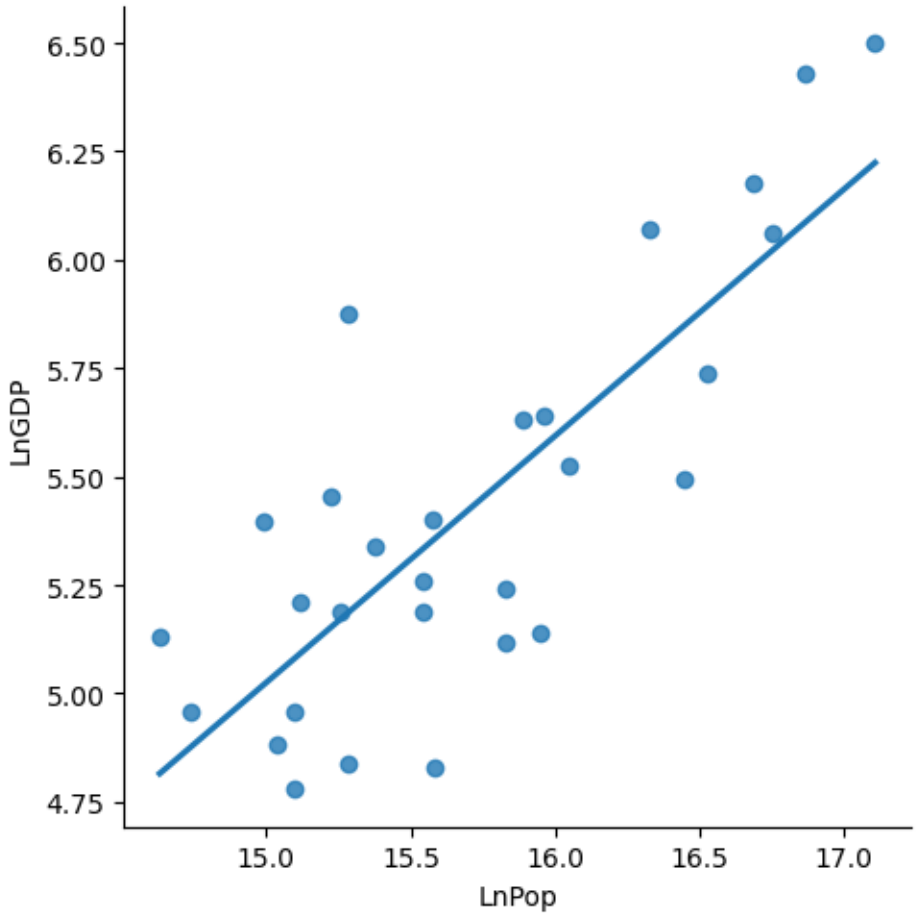
```
chinagdp = pd.read_excel('chinaGDP.xlsx')
chinagdp.head()
```

```
        City       Pop  GDP
0   Shanghai  26875500  664
1    Beijing  21167303  619
2   Shenzhen  17633800  482
3  Chongqing  12313714  433
4  Guangzhou  18810600  429
```

```
for i in chinagdp.index:
    chinagdp.loc[i,"LnPop"]=np.log( chinagdp.loc[i,"Pop"])
    chinagdp.loc[i,"LnGDP"]=np.log( chinagdp.loc[i,"GDP"])
chinagdp.shape
```

```
(29, 5)
```

```
sns.lmplot(
    data = chinagdp,
    x = 'LnPop',
    y = 'LnGDP',
    ci = None
);
```




```
x = 'LnPop'
y = 'LnGDP'

lin_mod = sm.regression.linear_model.OLS(
    chinagdp[y], # the dependent variable (y)
    sm.tools.tools.add_constant(chinagdp[x]) # the independent variable (x)
)

results = lin_mod.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  LnGDP   R-squared:                       0.638
Model:                            OLS   Adj. R-squared:                  0.624
Method:                 Least Squares   F-statistic:                     47.51
Date:                Fri, 29 Dec 2023   Prob (F-statistic):           2.09e-07
Time:                        11:03:43   Log-Likelihood:                -4.2813
No. Observations:                  29   AIC:                             12.56
Df Residuals:                      27   BIC:                             15.30
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
```

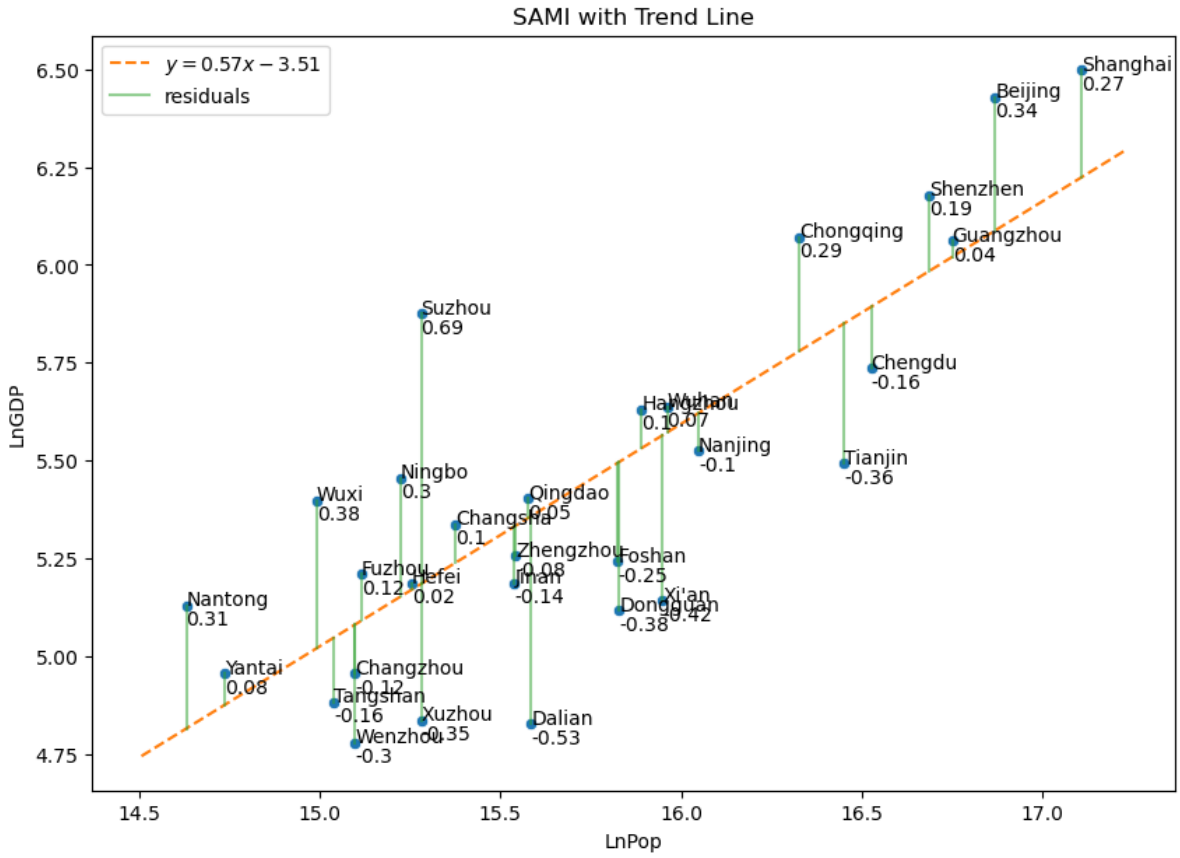
(continues on next page)

(continued from previous page)

```
------------------------------------------------------------------------------
const         -3.5082      1.298     -2.703      0.012      -6.171      -0.846
LnPop          0.5689      0.083      6.893      0.000       0.400       0.738
==============================================================================
Omnibus:                        0.209   Durbin-Watson:                   1.699
Prob(Omnibus):                  0.901   Jarque-Bera (JB):                0.324
Skew:                           0.176   Prob(JB):                        0.850
Kurtosis:                       2.620   Cond. No.                         380.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly␣
↪specified.
```

Note that the data for China has β=.5689.

```
Text(0.5, 1.0, 'SAMI with Trend Line')
```



Exercise

Exercise

Where would Hong Kong fall if added to the SAMI graph above?

15.6.5 Internally Displaced People (IDP) in Tigray, Ethiopia

A peace agreement was signed in Pretoria, South Africa in November 2022, two years after a horrific war broke out in Tigray, the northernmost region of Ethiopia. An estimated four hundreds thousand Tigrayan civilians were killed out of a population of 6 or 7 million, and a hundred thousand women were raped. The entire region was traumatized. Thousands of schools were damaged/closed and millions of children were out of school for over two years. The vast majority of medical facilities were rendered inoperative. Young people in large number escaping reality through drugs and alcohol additions.

One year after the peace agreement, fighting continued in western Tigray and 1 million internally displaced people (IDP) were living in over 600 IDP camps scattered across Tigray. Most camps are have sub-standard living conditions–acute food and clean water shortages; many living in severely overcrowded rooms in damaged schools or in tents; no livelihood, nothing to return to even if peace is established back home.

In this lab, we look at IDP camp data collected in summer 2023. The data set is large (638 rows or camp observations and 468 columns or attributes) so we begin with exploratory data analysis to get a feel for the data as a whole. Our goal then shifts to addressing the question of how an NGO concerned with the plight of IDPs might prioritize their efforts based on i) areas within Tigray that are underserved and ii) ranking of the severity of the need of individual camps within underserved regions.

The IDP data we analyzed was obtained at https://data.humdata.org/dataset/ethiopia-displacement-northern-region-tigray-idps-site-assessment-iom-dtm?

1. Let's put the IDP data into a dataframe called "df".

```
df = pd.read_excel("IDP.xlsx")
df.head(1)
```

```
  1.1.a.1: Survey Date        Country   Country Code   Reported Date  \
0       #date+occurred  #country+name  #country+code  #date+reported

  1.1.a.2: Survey Round 1.1.c.1: Site ID 1.1.d.1: Site Name  \
0                   NaN              NaN                NaN

  1.1.d.2: Site Alternate Name 1.4.a.2: Is site open? 1.1.e.1: Region  ...  \
0                          NaN                    NaN      #adm1+name  ...

  S1723: Other internet sources (e.g. apps)  \
0                                       NaN

  S1723: Please specify which other internet sources S1723: Other  \
0                                                NaN          NaN

  S1723: If other source of news/information, please specify  \
0                                                NaN

  S1784: Is mobile network access available in the site?  \
0                                                NaN
```

(continues on next page)

(continued from previous page)

```
  S1495: What % of HHs own a mobile phone?  \
0                                      NaN

  11.3.a.1: Are members of the community discussing/advertising travel␣
↪opportunities?  \
0                                                 NaN

  11.3.a.6: If Yes, to where? 11.3.a.2: Specify all locations  \
0                          NaN                              NaN

  M1712: Additional Comments / Observations
0                                       NaN

[1 rows x 488 columns]
```

2) Select columns to assess disorder.

```
             Name     Zone Woreda     Long       Lat    Pop Crowded  \
1  Hareze Seb'ata  Eastern   Erob  39.5828  14.43946  10410    High

                     Registration Food Shelter WASH Livelihood Healthcare  \
1  Yes, but the list is incomplete   No     Yes  Yes        Yes         No

  Protection Education
1         No        No
```

```
data.shape
```

```
(642, 15)
```

3) Binary code deficiencies (1=deficient)

```
             Name           Zone     Woreda      Long       Lat    Pop  \
1  Hareze Seb'ata        Eastern       Erob  39.58280  14.43946  10410
2       Enda Mosa        Eastern       Erob  39.55493  14.42384  10230
3      Adi Abagie  North Western  Adi Daero  38.18660  14.26590    196
4      May Ambssa  North Western  Adi Daero  38.23350  14.22740    117
5          Hibret  North Western  Adi Daero  38.16930  14.32020    242

   Crowded  Registration  Food  Shelter  WASH  Livelihood  Healthcare  \
1        1             0     0        1     1           1           0
2        1             0     1        1     0           1           0
3        0             1     1        0     0           0           0
4        0             1     1        0     0           0           0
5        0             1     1        1     0           0           0

   Protection  Education
1           0          1
2           0          1
3           0          1
4           0          1
5           0          1
```

4) Find the total number of needs in each camp and the total number of IDP in camps with i needs.

```
for i in data.index:
    data.loc[i,"sum"]=int(data.loc[i,'Crowded']+data.loc[i,'Registration']+data.loc[i,
 ↪'Food']+data.loc[i,'Shelter']+data.loc[i,'WASH']+data.loc[i,'Livelihood']+data.
 ↪loc[i,'Healthcare']+data.loc[i,'Protection']+data.loc[i,'Education'])
data.head(3)
```

```
             Name           Zone     Woreda      Long       Lat    Pop  \
1  Hareze Seb'ata        Eastern       Erob  39.58280  14.43946  10410
2       Enda Mosa        Eastern       Erob  39.55493  14.42384  10230
3      Adi Abagie  North Western  Adi Daero  38.18660  14.26590    196

   Crowded  Registration  Food  Shelter  WASH  Livelihood  Healthcare  \
1        1             0     0        1     1           1           0
2        1             0     1        1     0           1           0
3        0             1     1        0     0           0           0

   Protection  Education  sum
1           0          1  5.0
2           0          1  5.0
3           0          1  3.0
```

```
totalpop=population(data)
np.sum(totalpop)
```

```
1021593
```

5) Create dataframes with data for each of the 6 zones.

```
data["Zone"].value_counts()
```

```
Central          191
North Western    140
Eastern          130
South East        73
Mekelle           55
Southern          53
Name: Zone, dtype: int64
```

```
tot=642  #total number of camps
```

```
U=data["sum"].sum()/tot
U  #mean number of needs
```

```
3.1183800623052957
```

```
central_df = data[data['Zone'] == 'Central']
centralpop=needlevel(central_df)
central_df.head(1)
```

```
      Name     Zone          Woreda    Long      Lat  Pop  Crowded  \
9  Kewanit  Central  Tahtay Maychew  38.634  14.0922  136        0

   Registration  Food  Shelter  WASH  Livelihood  Healthcare  Protection  \
9             0     1        0     1           0           1           0
```

(continues on next page)

(continued from previous page)

```
   Education  sum
9          1  4.0
```

```
centralpop
```

```
[0, 185, 66443, 98460, 34418, 24262, 0, 0, 0, 0]
```

```
NW_df = data[data['Zone'] == 'North Western']
NWpop=needlevel(NW_df)
NW_df.head(1)
```

```
         Name           Zone     Woreda     Long      Lat  Pop  Crowded  \
3  Adi Abagie  North Western  Adi Daero  38.1866  14.2659  196        0

   Registration  Food  Shelter  WASH  Livelihood  Healthcare  Protection  \
3             1     1        0     0           0           0           0

   Education  sum
3          1  3.0
```

```
NWpop
```

```
[0, 5917, 84960, 105586, 78965, 34967, 5242, 0, 0, 0]
```

```
eastern_df = data[data['Zone'] == 'Eastern']
easternpop=needlevel(eastern_df)
eastern_df.head(1)
```

```
             Name     Zone Woreda     Long       Lat    Pop  Crowded  \
1  Hareze Seb'ata  Eastern   Erob  39.5828  14.43946  10410        1

   Registration  Food  Shelter  WASH  Livelihood  Healthcare  Protection  \
1             0     0        1     1           1           0           0

   Education  sum
1          1  5.0
```

```
easternpop
```

```
[0, 702, 44725, 47223, 14953, 24134, 23113, 0, 0, 0]
```

```
# Filter rows with 'Mekele' in the 'City' column
SE_df = data[data['Zone'] == 'South East']
SEpop=needlevel(SE_df)
SE_df.head(1)
```

```
         Name        Zone   Woreda    Long      Lat  Pop  Crowded  \
263  Adi-Keyh  South East  Wejerat  39.667  13.0126  326        0

     Registration  Food  Shelter  WASH  Livelihood  Healthcare  Protection  \
263             0     1        0     0           0           0           0
```

(continues on next page)

(continued from previous page)

```
     Education  sum
263          1  2.0
```

```
SEpop
```

```
[0, 469, 11772, 9957, 12005, 0, 0, 0, 0, 0]
```

```
mekelle_df = data[data['Zone'] == 'Mekelle']
mekellepop=needlevel(mekelle_df)
mekelle_df.head(1)
```

```
                          Name     Zone            Woreda      Long       Lat  \
559  Hayelom Elementary School  Mekelle  Hawelti Sub City  39.45346  13.50069

      Pop  Crowded  Registration  Food  Shelter  WASH  Livelihood  Healthcare  \
559  1760        0             1     1        0     0           1           0

     Protection  Education  sum
559           0          0  3.0
```

```
mekellepop
```

```
[0, 14009, 79426, 78206, 29365, 33478, 0, 0, 0, 0]
```

```
southern_df = data[data['Zone'] == 'Southern']
southernpop=needlevel(southern_df)
southern_df.head(1)
```

```
    Name      Zone      Woreda     Long      Lat  Pop  Crowded  Registration  \
295  Ebo  Southern  Raya Azebo  39.6911  12.8469  123        0             0

     Food  Shelter  WASH  Livelihood  Healthcare  Protection  Education  sum
295     1        1     0           1           0           0          1  4.0
```

```
southernpop
```

```
[0, 0, 3203, 10645, 33313, 11490, 0, 0, 0, 0]
```

15.6.6 Exercises

Exercises

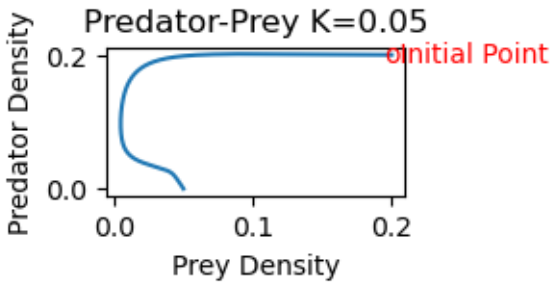
1. Compute the mean, weighted mean, entropy, and weighted entropy for each of the 6 zones.
2. Make a histogram for each zone which shows the number of IDP living in camps with i deficiencies (i=0,1,2,3,4,5,6,7,8,9).

15.7 JNB Lab Solutions

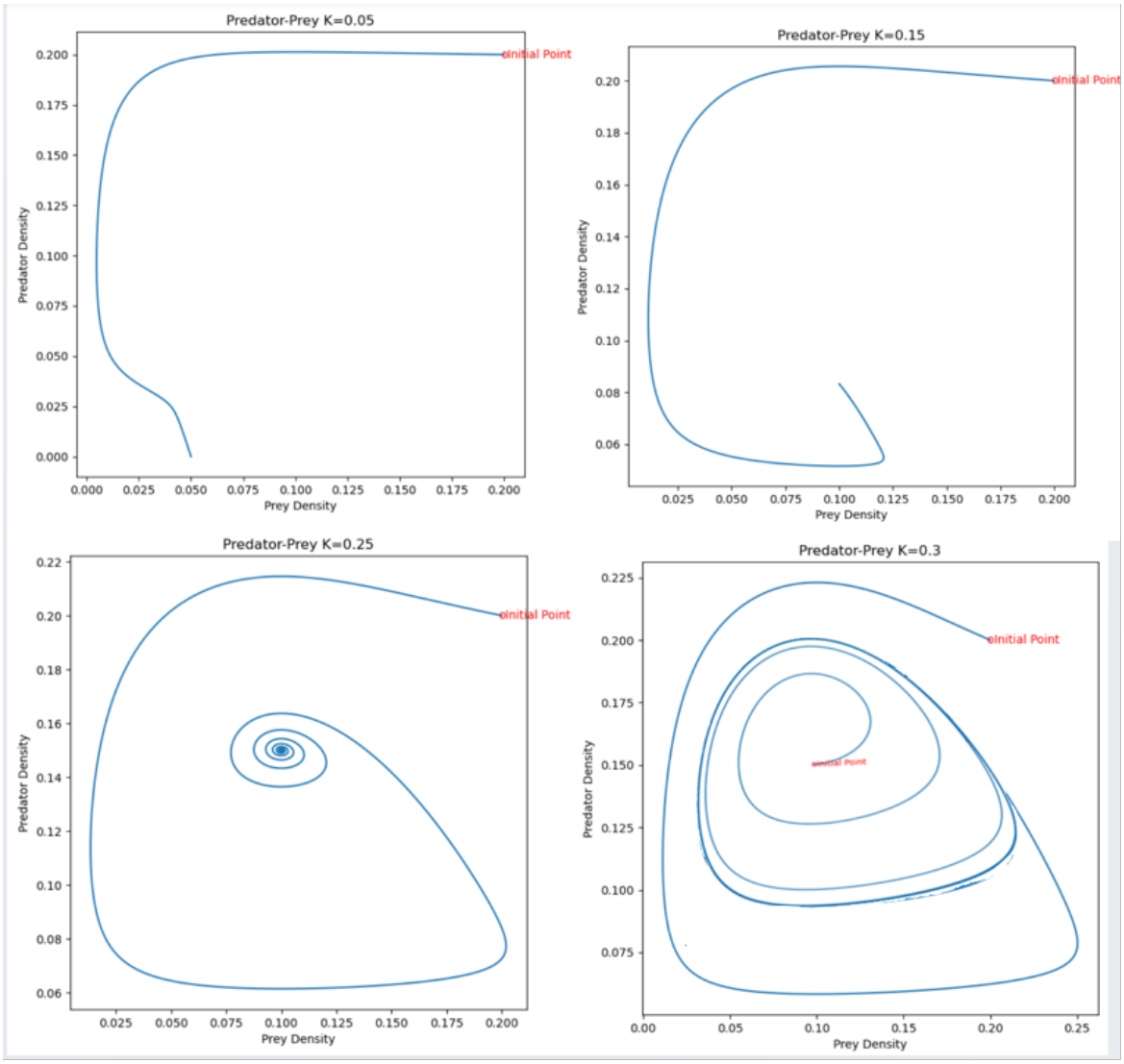
15.7.1 Bifurcations

Solution to Problem 1

```
# Probkem 1b
rmpp(.2,.2,.05)
```

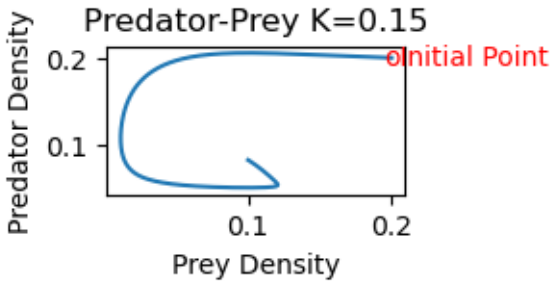


Solution to Problem 2a)

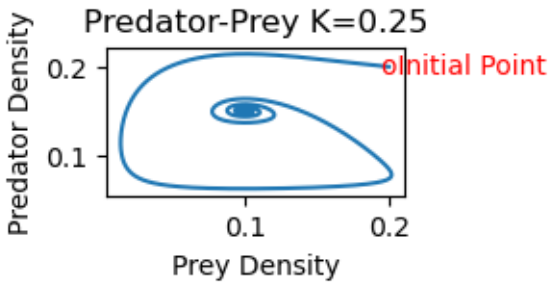


2b) As K increases, the equilibrium point shifts stability from one with zero predator concentration to one with positive predator concentration. Increasing K further, the equilibrium is still stable, but the solution spirals around the equilibrium point on its approach toward the equilibrium values. The steady state is hence also referred to as a stable spiral. Finally, as K increases still further,the equilibrium loses its stability and solutions approach a stable limit cycle (periodic solution) surrounding the equilibrium point.

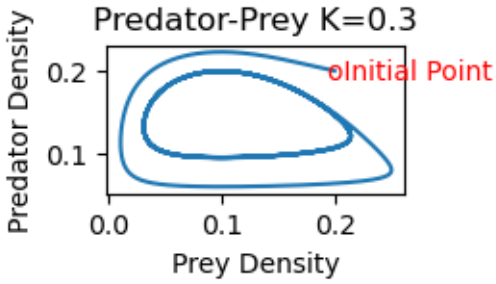
```
rmpp(.2,.2,.15)
```



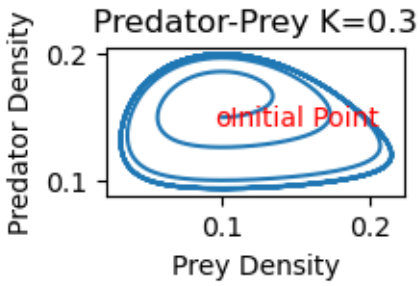
```
rmpp(.2,.2,.25)
```




```
rmpp(.2,.2,.3)
```




```
rmpp(.1,.15,.3)
```



Solution to Problem 3

The transition of the steady state with positive densities of both prey and predator from stable to unstable occurs in what is called a Hopf bifurcation point. As explained in the previous section at larger values of K the prey-predator equilibrium loses its stability while at the same time a limit cycle emerges, that is a closed loop of prey and predator densities which all timeseries of dynamics of the model tend to approach.For more information on this: https://staff.fnwi.uva.nl/a.m.deroos/projects/QuantitativeBiology/43-HopfPoint-Rosenzweig.html

15.7.2 Sensitive Dependence on Initial Conditions

Solution to problem 1

The FHN bursting behavior is related to Hopf bifurcation in the 2D system. The interval quiescense (burst pattern 00000) occurs at a subcritical parameter values where the equilibrium is a stable spiral. The two large spikes(11) occur for parameter values across the Hopf bifurcation point due to a stable periodic solution in the underlying 2D system

Solution to Problem 2

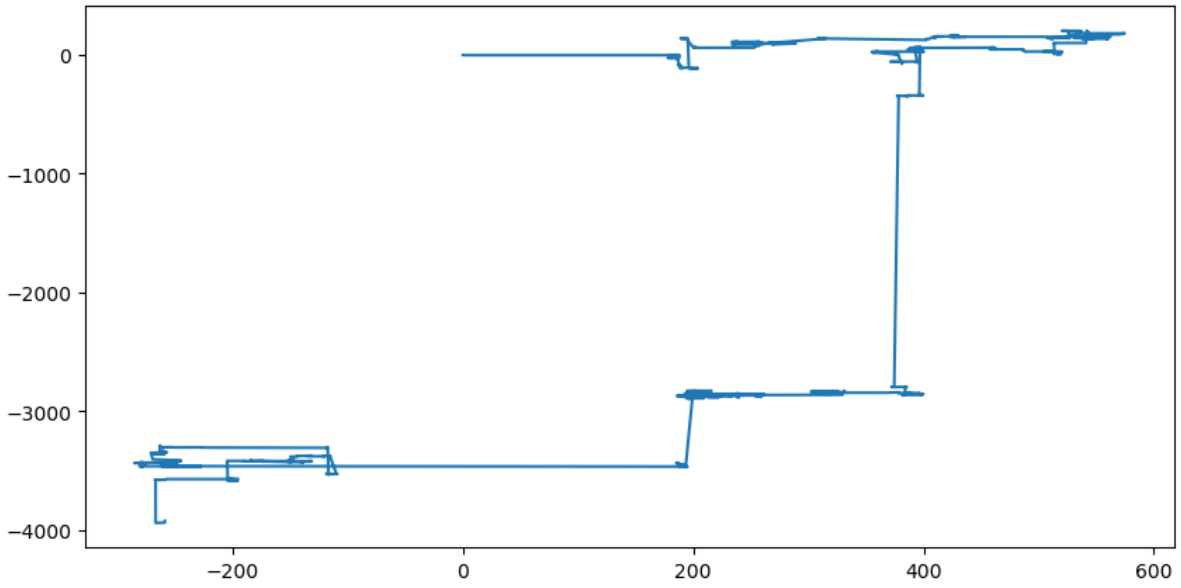
a) After plugging in x0=[1.725,.337,.00764] for the initial condition, the burst pattern changes to 11000101000000

b) Sensitive dependence occurs in non-linear systems where the initial conditions have a large impact on the solution behavior. For the the FHN system, changing the initial condition only slightly $[1.744, .298, .00872] \rightarrow [1.725, .337, .00764]$ results in a completely different stable burst pattern.

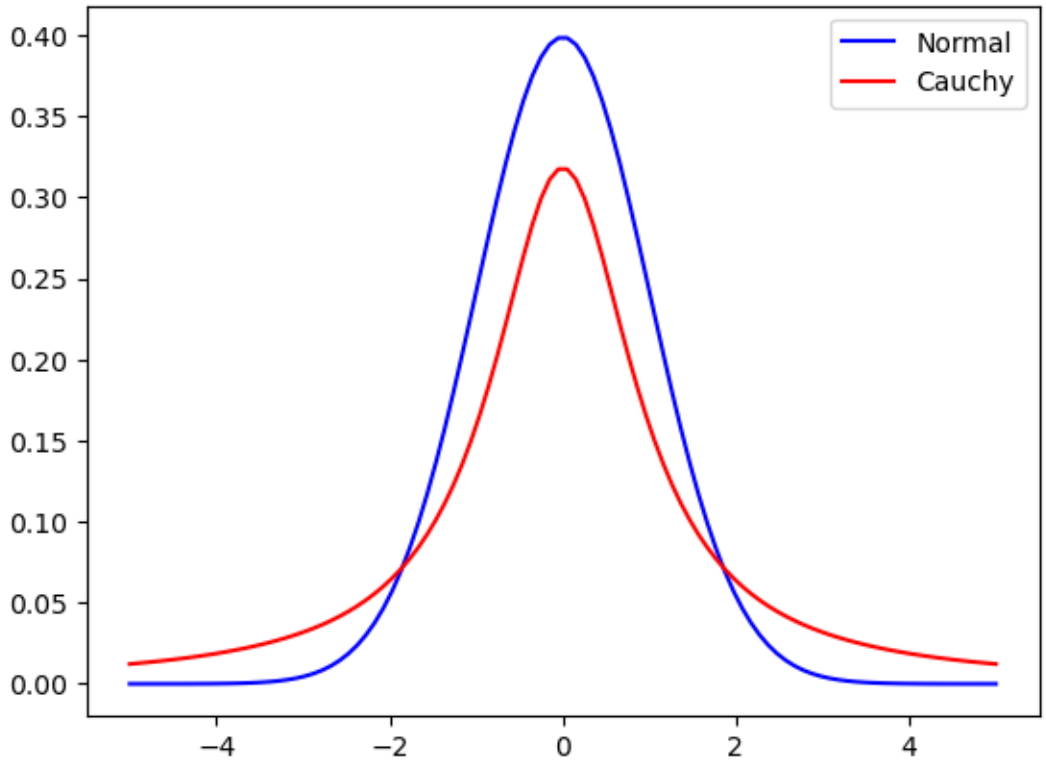
15.7.3 Random Walks

Solution to problem 1

a)



b) Unlike Brownian motion, a Cauchy distribution has long “flights.”



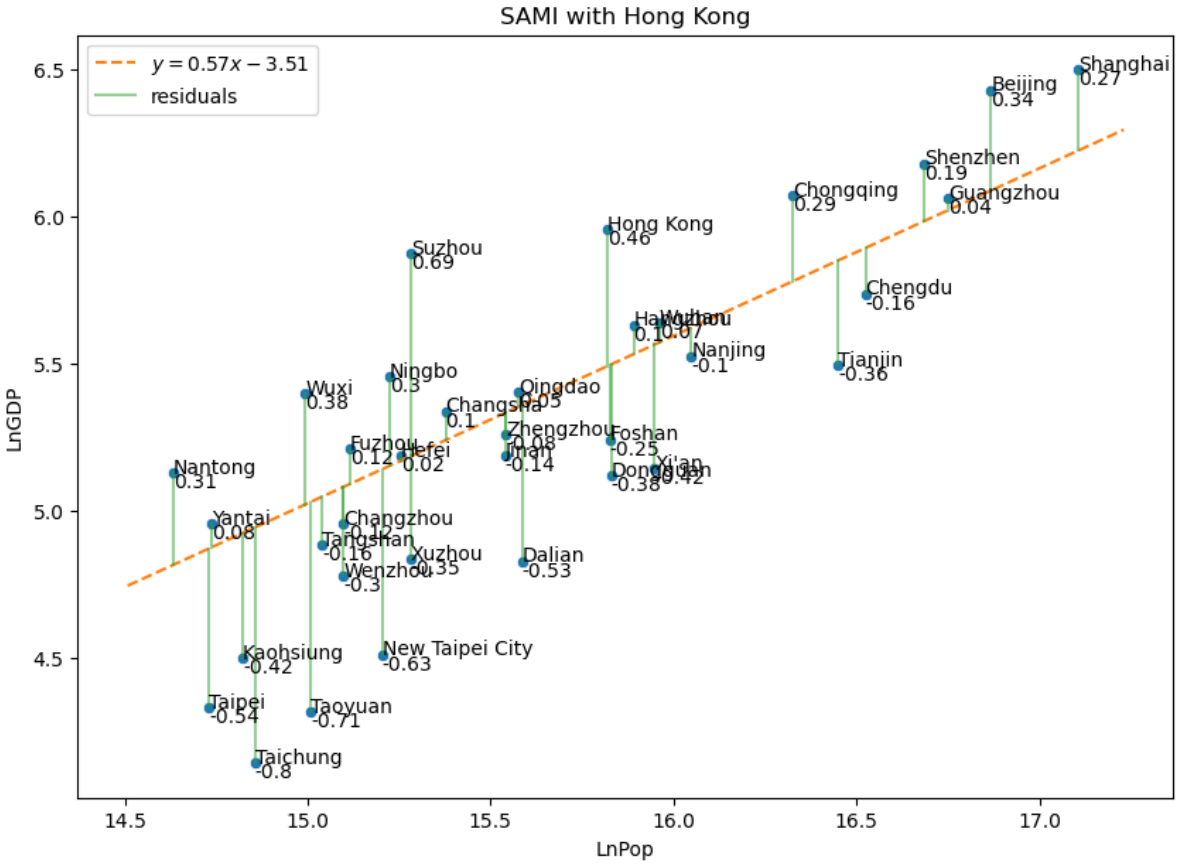
2. a) Unlike a normal distribution with standard deviation s.d.=1, the Cauchy distribution (in red) wih a heavy tail has a significant probability for random draws more than 3 or less than -3. That is what causes the long flights in 1a).

b) In some contexts, long flights are catastrophic events (eg. stock market crash).

15.7.4 Scale Adjusted Metropolitan Index

Solution to Exercise

```
Text(0.5, 1.0, 'SAMI with Hong Kong')
```



15.7.5 IDP

Solution to Problem 1

1. Read in IDP data from the file “idp.xlsx”

```
  1.1.a.1: Survey Date        Country   Country Code   Reported Date  \
0       #date+occurred  #country+name  #country+code  #date+reported

  1.1.a.2: Survey Round 1.1.c.1: Site ID 1.1.d.1: Site Name  \
0                   NaN              NaN                NaN

  1.1.d.2: Site Alternate Name 1.4.a.2: Is site open? 1.1.e.1: Region  ...  \
0                          NaN                   NaN      #adm1+name  ...

  S1723: Other internet sources (e.g. apps)  \
0                                       NaN

  S1723: Please specify which other internet sources S1723: Other  \
0                                                NaN          NaN

  S1723: If other source of news/information, please specify  \
0                                                NaN

  S1784: Is mobile network access available in the site?  \
0                                                NaN
```

(continues on next page)

(continued from previous page)

```
  S1495: What % of HHs own a mobile phone?  \
0                                      NaN

  11.3.a.1: Are members of the community discussing/advertising travel␣
↪opportunities?  \
0                                                NaN

  11.3.a.6: If Yes, to where? 11.3.a.2: Specify all locations  \
0                          NaN                              NaN

  M1712: Additional Comments / Observations
0                                       NaN

[1 rows x 488 columns]
```

2. Select columns to assess disorder.

```
             Name     Zone Woreda     Long       Lat    Pop Crowded  \
1  Hareze Seb'ata  Eastern   Erob  39.5828  14.43946  10410    High

                     Registration Food Shelter WASH Livelihood Healthcare  \
1  Yes, but the list is incomplete   No     Yes  Yes        Yes         No

  Protection Education
1         No        No
```

```
data.shape
```

```
(642, 15)
```

3) Binary code deficiencies (1=deficient)

```
             Name           Zone     Woreda      Long       Lat    Pop  \
1  Hareze Seb'ata        Eastern       Erob  39.58280  14.43946  10410
2       Enda Mosa        Eastern       Erob  39.55493  14.42384  10230
3      Adi Abagie  North Western  Adi Daero  38.18660  14.26590    196
4      May Ambssa  North Western  Adi Daero  38.23350  14.22740    117
5          Hibret  North Western  Adi Daero  38.16930  14.32020    242

   Crowded  Registration  Food  Shelter  WASH  Livelihood  Healthcare  \
1        1             0     0        1     1           1           0
2        1             0     1        1     0           1           0
3        0             1     1        0     0           0           0
4        0             1     1        0     0           0           0
5        0             1     1        1     0           0           0

   Protection  Education
1           0          1
2           0          1
3           0          1
4           0          1
5           0          1
```

4) Find the total number of needs in each camp and the total number of IDP pop_i in camps with i needs.

```
for i in data.index:
    data.loc[i,"sum"]=int(data.loc[i,'Crowded']+data.loc[i,'Registration']+data.loc[i,
 ↪'Food']+data.loc[i,'Shelter']+data.loc[i,'WASH']+data.loc[i,'Livelihood']+data.
 ↪loc[i,'Healthcare']+data.loc[i,'Protection']+data.loc[i,'Education'])
data.head(3)
```

```
             Name           Zone     Woreda      Long       Lat    Pop  \
1  Hareze Seb'ata        Eastern       Erob  39.58280  14.43946  10410
2       Enda Mosa        Eastern       Erob  39.55493  14.42384  10230
3      Adi Abagie  North Western  Adi Daero  38.18660  14.26590    196

   Crowded  Registration  Food  Shelter  WASH  Livelihood  Healthcare  \
1        1             0     0        1     1           1           0
2        1             0     1        1     0           1           0
3        0             1     1        0     0           0           0

   Protection  Education  sum
1           0          1  5.0
2           0          1  5.0
3           0          1  3.0
```

```
totalpop=needlevel(data)
np.sum(totalpop)
```

```
1021593
```

```
      Name     Zone          Woreda    Long      Lat  Pop  Crowded  \
9  Kewanit  Central  Tahtay Maychew  38.634  14.0922  136        0

   Registration  Food  Shelter  WASH  Livelihood  Healthcare  Protection  \
9             0     1        0     1           0           1           0

   Education  sum
9          1  4.0
```

```
centralpop
```

```
[0, 185, 66443, 98460, 34418, 24262, 0, 0, 0, 0]
```

```
         Name           Zone     Woreda     Long      Lat  Pop  Crowded  \
3  Adi Abagie  North Western  Adi Daero  38.1866  14.2659  196        0

   Registration  Food  Shelter  WASH  Livelihood  Healthcare  Protection  \
3             1     1        0     0           0           0           0

   Education  sum
3          1  3.0
```

```
NWpop
```

```
[0, 5917, 84960, 105586, 78965, 34967, 5242, 0, 0, 0]
```

```
             Name     Zone Woreda     Long       Lat    Pop  Crowded  \
1  Hareze Seb'ata  Eastern   Erob  39.5828  14.43946  10410        1
```

(continues on next page)

(continued from previous page)

```
   Registration  Food  Shelter  WASH  Livelihood  Healthcare  Protection  \
1             0     0        1     1           1           0           0

   Education  sum
1          1  5.0
```

```
easternpop
```

```
[0, 702, 44725, 47223, 14953, 24134, 23113, 0, 0, 0]
```

```
         Name        Zone   Woreda    Long      Lat  Pop  Crowded  \
263  Adi-Keyh  South East  Wejerat  39.667  13.0126  326        0

     Registration  Food  Shelter  WASH  Livelihood  Healthcare  Protection  \
263             0     1        0     0           0           0           0

     Education  sum
263          1  2.0
```

```
SEpop
```

```
[0, 469, 11772, 9957, 12005, 0, 0, 0, 0, 0]
```

```
                          Name     Zone            Woreda      Long       Lat  \
559  Hayelom Elementary School  Mekelle  Hawelti Sub City  39.45346  13.50069

      Pop  Crowded  Registration  Food  Shelter  WASH  Livelihood  Healthcare  \
559  1760        0             1     1        0     0           1           0

     Protection  Education  sum
559           0          0  3.0
```

```
mekellepop
```

```
[0, 14009, 79426, 78206, 29365, 33478, 0, 0, 0, 0]
```

```
    Name      Zone      Woreda     Long      Lat  Pop  Crowded  Registration  \
295  Ebo  Southern  Raya Azebo  39.6911  12.8469  123        0             0

     Food  Shelter  WASH  Livelihood  Healthcare  Protection  Education  sum
295     1        1     0           1           0           0          1  4.0
```

```
southernpop
```

```
[0, 0, 3203, 10645, 33313, 11490, 0, 0, 0, 0]
```

5) Compute the mean, weighted mean, entropy, and deficiency entropy for each zone.

```
Central Zone Entropy= 1.256
Central Zone Mean= 3.072
Central Zone Weighted Mean= 0.199
Central Zone Deficiency Entropy= 0.219
```

```
NW Zone Entropy= 1.453
NW Zone Mean= 3.215
NW Zone Weighted Mean= 0.245
NW Zone Deficiency Entropy= 0.324
```

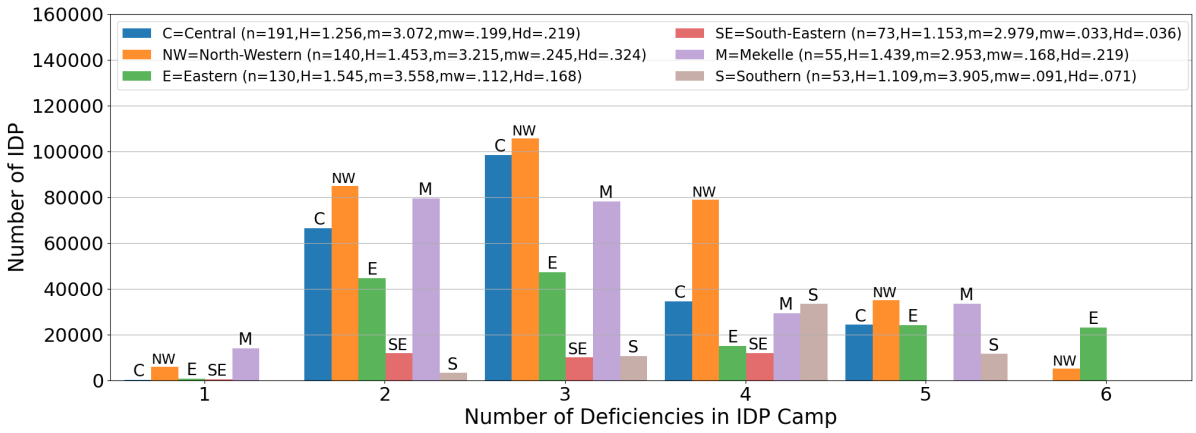
```
Eastern Zone Entropy= 1.545
Eastern Zone Mean= 3.558
Eastern Zone Weighted Mean= 0.112
Eastern Zone Deficiency Entropy= 0.168
```

```
SE Zone Entropy= 1.153
SE Zone Mean= 2.979
SE Zone Weighted Mean= 0.033
SE Zone Deficiency Entropy= 0.036
```

```
Mekelle Zone Entropy= 1.439
Mekelle Zone Mean= 2.953
Mekelle Zone Weighted Mean= 0.168
Mekelle Zone Deficiency Entropy= 0.219
```

```
Southern Zone Entropy= 1.109
Southern Zone Mean= 3.905
Southern Zone Weighted Mean= 0.091
Southern Zone Deficiency Entropy= 0.071
```

Solution to Problem 2



15.7.6 References

Bertin, E. 2021. *Statistical Physics of Complex Systems: A Concise Introduction (3rd Edition)*. Cham, Switzerland: Springer Nature.

Bettencourt, L. 2022. *Introduction to Urban Science: Evidence and Theory of Cities as Complex Systems.* Cambridge, MA: The MIT Press.

Fieguth, P. 2021.*An Introduction to Complex Systems: Society, Ecology, and Nonlinear Dynamics (2nd Edition)*. Cham, Switzerland: Springer Nature.

Ghosh, A. 2017. *Dynamic Systems for Everyone: Understanding how our world works. (2E)* Cham, Switzerland: Springer.

Isihara P, Diedrichs DR, Kartawijaya T, Townsend E, and Tully K. 2017. Disaster Relief Modeling, The UMAP Journal, Vol. 38(4):399-430.

Macauley, T. 2017. RIoT Control Understanding and Managing Risks and the Internet of Things. Amsterdam: Elsevier.

Martin, B. and de Roos, A. 2023. Hopf bifurcation in a predator-prey model. https://staff.fnwi.uva.nl/a.m.deroos/projects/QuantitativeBiology/43-HopfPoint-Rosenzweig.html

Murray, J.D. 2002. *Mathematical Biology: I. An Introduction (3E).* New York: Springer.

Sampson, R. 2012. *Great American City: Chicago and the Enduring Neighborhood Effect.* Chicago: The University of Chicago Press.

Siegenfeld, A. and Bar-Yam, Y. 2020. An introduction to complex system science and its applications. https://www.hindawi.com/journals/complexity/2020/6105872/

Tranquillo, J. 2019. An Introduction to Complex Systems: Making Sense of a Changing World. Cham, Switzerland: Springer Nature.

Index

absolute extrema, 476, 531
acceleration, 907
accuracy, 66
algorithms, 106
alignment, 1120
Allen NLP NER, 1021
alpha values, 48
Alzheimer's disease, 891
analysis of variance (*ANOVA*), 168, 201
animation, 64, 81, 96, 396, 538, 544
antiderivative, 483
aquifer, 989
area between two curves, 409, 495
area problems, 486
area under the curve (*AUC*), 1048
autonomous equation, 790
average cost, 381
average rate of change, 465
average value, 493, 534

bag of words, 1025
bar chart, 194
barchart, 135
basis, 600, 698
Bayes' Theorem, 1044
Beautifulsoup, 1053
Bernoulli's equation, 904
best approximation theorem, 645
bifurcation, 1124, 1155, 1169
binary classification of labeled data, 732
block descent, 723
box plot, 166, 220, 226
boxplot, 196
break-pressure tank, 911
Brownian motion, 1127, 1160, 1172

canonical ensemble, 1131
catastrophic event, 1127
categorical variable, 161
Cauchy distribution, 1127, 1160, 1172
Cauchy-Riemann equation, 977
Cauchy-Shwarz inequality, 636
census tracts, 217, 226
central limit theorem, 296
chain rule, 472, 976
change of basis, 627
characteristic polynomial, 683
characters, 1017
chi-squared test, 204
Chicago Data Portal, 215
circuits, 26
City of Chicago budget, 82
class, 63
classes, 111, 151, 154
classification, 1041
classification of texts, 1041
cleaning data, 158
closed system, 1119
Cobbs-Douglas Production, 387
cofactor expansion, 587
coherent system, 1119
cohesion, 1120
collective efficacy, 1117
column matrix, 556
column space, 596, 698
compiler language, 174
complex eigenvalues in ODE systems, 807
complex functions, 974
complex logarithms, 975
complex numbers, 974
complex potential, 983, 998, 1005
complex system science, 1117, 1155, 1169
concave down, 454, 531
concave up, 454, 531
conditionals, 100
confidence intervals, 302, 356, 376
consistent, 550, 696
contaminant flow, 998, 1005
continuity, 464, 525

continuity equation, 981
continuous compounding interest, 412
control volume, 907
convergence and divergence of series, 515
convergence tests, 517
convergent, 412
coordinate descent, 723
corpus, 1017
correlated system, 1119
correlation coefficient, 340
covariance matrix, 727
COVID-19, 38, 881
critical value method, 992
cumulative distribution function (*CDF*), 442
cv2, 53
CWS model, 893
cycloid, 543
cylindrical coordinates, 24

Darcy's Law, 982
data likelihood, 1044
data operations, 119
data visualization, 288
dataframe, 35, 127
dataframe attributes, 177
decreasing, 454
definite integrals, 486
demand function, 381
Department of Human Services (*DHS*), 172
derivative, 469, 528
derivative optimization, 437, 448
derivative rules, 380
derivatives of trig functions, 392
designing a water-delivery system, 922
determinant of matrix, 586, 697
determinants, 568, 697
determinants as area or volume, 593, 697
deterministic system, 1127
diagonal entry, 569
diagonal matrix, 569
diagonalizable, 622
diagonalization, 622, 684
difference of two proportions, 314
differential equations, 779
dimension, 600, 698
dimension reduction by principal component analysis, 725
direction fields, 786
discrete dynamical systems, 628, 687
display
    image, 4
    youtube video, 4
distribution, 397
divergent, 412
document embedding, 1036
document-term matrices, 1028
domain, 453, 523
domain knowledge, 173
dummy variables, 351

echelon form, 552, 696
EDA reports, 207, 213
Eidge Classification, 1045
eigenspace, 619, 683, 698
eigenvalues, 615, 698, 804
eigenvectors, 615, 698
elementary row operations, 552, 696
email filtering, 1015
emergent behavior, 1120
energy, 1128
entropy, 437, 448, 1119, 1131, 1168, 1174
equilibrium analysis, 894, 962, 968
equilibrium bundle, 378
equilibrium point, 1121
equilibrium solution
    semi-stable, 791
    stable, 791
    unstable, 791
equilibrium systems, 1128
equipotentials, 983, 998, 1005
evaluating classification, 1046
exploratory data analysis (*EDA*), 155
    advanced, 172
exponential decay, 417
exponential distribution, 398, 399, 441, 450
exponential functions, 403, 975
exponential growth, 416, 881
expressions, 98
extreme value theorem, 477

F1 score, 66, 1034
false negative, 43
false positive, 43
false positive rate (*FPR*), 1048
feature effectiveness, 1034
features, 1033
Fischer-Calo, 973, 1000, 1007
Fitzhugh Nagumo systems, 1156, 1171
fixed cost, 381
flow rate, 981
fluid mechanics, 904
folium, 41, 217, 747
for loops, 33, 66
fractals, 25
fractions, 21
frequencies of words, 57
frequency, 58

frequency distribution, 289, 290
function compositions, 460
function operations, 460
functions, 28, 37, 101, 453
fundamental set of solutions, 803
fundamental theorem of calculus, 490

general solution, 803
generation, 187
geometric behavior of homogeneous linear ODE systems, 813
geometric multiplicity, 626, 698
GINI index, 411
gradient descent, 722
gradient-based optimization, 718
Gram-Schmidt process, 647, 692
grand-canonical ensemble, 1131
graphical intersections, 381
gravity-fed water delivery, 903
Great Migration, 44, 66
groundwater flow equation, 981

half-life, 417
hardship index, 45, 69
harmonic conjugate, 979
harmonic functions, 979
heat map, 45
Heaviside functions, 827
heavy tail, 1127
hiding code, 210
histogram, 160, 192, 211, 225, 289, 375, 1168, 1174
homicides, 287
homogeneity, 1119
homogenous, 989
Honduras, 923
Horner's rule, 149, 153
Housing Equity Initiative (*HEI*), 214
hydrology, 972
hypocycloid, 545
hypothesis, 309
hypothesis test, 306, 360
hypothesis testing, 357

identity matrix, 569
if conditional statements, 34, 66
illogical value removal, 189
image recognition, 761
implicit differentiation, 480
importing and exporting data, 119
improper integral, 492
income streams, 414
income-consumption path, 378
incompressible, 910
inconsistent, 550, 696
increasing, 454
indefinite integrals, 483, 533
inflection points, 474
initial condition, 415
initial value problem (*IVP*), 415
inline (lambda) functions, 102
inner join, 44
inner product, 633, 688, 698
installing libraries, 117
instantaneous rate of change, 465
integral test, 412
integrals, 482
integration by parts, 485
intent detection, 1015
Intermediate Value Theorem, 465
internally displaced people (*IDP*), 1116
International Office for Migration (*IOM*), 1116
interpreter language, 174
inverse document frequency, 1033
inverse functions, 461
inverse Laplace transforms, 825
inverse of a square matrix, 580, 671, 697
inviscid flow, 910
Ising spin model, 1131
isolated paramagnetic spin model, 1131
isolated system, 1119
isotropic, 989

Jacobian matrix, 1121
JNB, 27
joint probability, 1032

k-means clustering, 47, 219, 720, 742, 747, 769
keyword extraction, 1015

L'Hospital's rule, 473, 976
laminar flow, 914
language detection, 1015
Laplace equation, 979
Laplace transforms, 823
leading entry, 552
least-squares problems, 654, 694, 698, 715
level curve, 9
levels of processing, 1017
libraries, 110
limits, 463, 525
linear approximation, 908
linear combination, 556, 557
linear dependence, 558
linear equations, 550, 696
linear first-order ODE, 783
linear homogeneous differential equations with constant coefficients, 794
linear independence, 558

linear regression, 332
  simple models, 334
linear systems, 378, 550, 661, 696
linear transformations, 595, 604, 681, 698
linearity, 976
linearization, 479, 532, 1121
lists, 103
logistic growth, 881
logistic growth model, 883, 1121
Lorentz curve, 410

machine learning, 1041
magnitude of vectors, 633
map, 40
marginal cost, 380
marginal functions, 380
markdown, 209
mathematical formalism, 735
matplotlib, 32, 131, 160
matrices, 568, 697
matrix algebra, 568, 669, 697
matrix entry, 569, 697
matrix equations, 556, 560, 665
maximization of projected variance, 728
maximum likelihood, 1134
mean, 403
mean value theorem, 478, 494, 532
microcanonical ensemble, 1131
migration matrix, 628
minimizing the OLS loss function via normal equations, 717
missing data, 45
Monte Carlo method, 289
Monte Carlo simulation, 291, 439, 449
multiclass classification, 1048
music, 80, 97
musical tone, 91

N/A value, 158
n-grams, 1031
Naive Bayes algorithms, 1034, 1044
named entity recognition (*NER*), 1021
named entity resolution, 1015
NaN values, 180
natural exponential functions, 456
natural language generation, 1033
natural language processing, 1015
Newton's 2nd Law, 903
NFL player positions, 85
non-deterministic system, 1127
non-homogeneous linear ODE systems with constant coefficients, 808
nonhomogeneous linear differential equations, 796
nonlinear systems, 1121
nonzero vectors, 639, 698
normal distribution, 403
normal equations, 654, 698, 741
normality test, 206
normalization, 1044
null distribution, 311
null hypothesis, 167, 200
null space, 596, 698
numeric count vectors, 1028
numpy, 29, 115

objects, 111
OLS linear regression, 7, 363, 715, 745, 769
one vs one (*OvO*), 1050
one vs rest (*OvR*), 1048
one-hot vectors, 1028
open system, 1119
opencv-python, 51
optimization, 383
optimization for linearly-separable data, 737
optimization for non-separable data, 737
optimization problem, 716
ordinary differential equation, 415, 781
  first order, 781
  separable, 781
orthogonal basis, 639, 698
orthogonal compliment, 637, 698
orthogonal projection, 639, 690, 698
orthogonal projections onto 1-dimensional spaces, 641
orthogonal sets, 639, 698
orthogonality, 633, 688, 698, 980
orthonormal basis, 646
os, 53
outlier removal, 189

p-series, 412
p-value, 167
pandas, 31, 65, 124, 127, 156
paragraphs, 1017
parameter estimation, 896
parametric curve, 8, 545
parametric curves, 543
parametric equations, 497, 534
Pascal's Triangle, 20
perceptron, 1042
Perelson's immunological HIV model, 960, 968
phase plane, 810
phase plane for nonlinear systems, 818
phase transitions, 1134
pie chart, 137

piechart, 83, 93
pipe selection, 920
pixel images, 48, 51, 75, 84, 94, 772, 773
polar coordinates, 23, 504
polynomial functions, 146, 153, 975
pooled sample proportion, 316
Porter stemmer, 1019
posterior, 1044
power law distribution, 1127
power rule, 976
power series, 521
precision, 43, 66, 1034
prediction, 66
preprocessing, 1015
principal component analysis, 728
principal component analysis (*PCA*), 743, 748, 772
prior, 1044
probabilistic language modeling, 1032
probability distributions, 438, 449
procedures, 1019
product rule, 472, 530, 976
projected variance, 727
projection, 606
pseudo-random number generator (*PRNG*), 399, 440, 450
pure tones, 58
python list comprehensions, 148, 153

quadric surface, 9
quotient rule, 472, 530, 976

radian measure, 392
random number simulation, 440, 450
random numbers, 61, 81
random process, 1127
random system, 1119
random walk, 1127, 1157, 1172
range, 453
rank, 600, 698
Rank-Nullity Theorem, 681
rates, 525
rational functions, 975
reading and writing data, 119
reading excel sheets, 83, 93
recall, 43, 66, 1034
recursive algorithms, 147, 153
redlining, 1117
reduced echelon form, 552, 696
reduction algorithm, 553
regression, 1041
    least-squares, 337
    multiple, 349
related rates, 391
relative extrema, 476, 531
residuals, 336
Reynold's number, 914
Riemann Sum, 488, 533, 540, 545
ROC Curve, 1048
row equivalent, 552
row reduction algorithm, 552, 696
Rozenzweig-MacArthur predator prey (RMPP) equations, 1155, 1169

scalar product, 556, 572, 697
scatter plot, 195
scatterplot, 7, 164
Schelling model, 1135
seaborn, 161
secant line, 466, 538, 544
semantic analysis, 1015
sensitivity, 43
sensitivity analysis, 379, 965, 968, 993
sentences, 1017
sentiment analysis, 1015
separating hyperplanes, 735
separation of variables, 415
sequences, 511, 537
series, 124, 438, 449, 514, 537
shape file, 215
signed distance, 736
similarity retrieval, 1057
simulation model, 397
sine waves, 58
sink, 11, 910, 985
slope fields, 785
solution set of linear systems, 667
sound waves, 89
source, 11, 910, 985, 999, 1005
space leasing plume, 990, 1002
span, 556
specificity, 43
spin system, 1131
spiral sink, 1124
spring mass problem, 1128
standard deviation, 403
standard matrix representation, 610, 698
statements, 98
statistical inference, 356
statistical physics, 1119, 1128
statistical process control, 403
statistical test, 167, 200
steady flow, 910
steady state, 1124
stemming, 1019
stream function, 983
streamline, 11, 905
streamline modeling, 988

string operations, 1022
Student's t-distribution, 322
subsets, 22
subspace, 595, 678, 698
substitution, 185
sum of matrices, 572
sum of squares error, 336
summary statistics, 46, 158, 170
superposition principle, 793
support vector machines (*SVM*), 733, 744, 757
symmetry, 24
syntactic analysis, 1015
system sensitivity method, 991
systems of first-order differential equations, 801
systems of linear equations, 550, 696

t-test, 167, 200, 323
    one-sample, 201
    paired, 201
    two-sample, 327, 374, 376
tangent line, 467, 538, 544
tax sale, 216, 224
test statistic, 310, 328
testing set, 762, 774, 1041
text classification, 1015
TF-IDF (term frequency, 1033
the Hamiltonian, 1128
The Rankine Oval, 986
Tigray, 1116
time-reversibility, 1128
tokenizers, 1020
total cost, 380
trace of square matrix, 585, 671, 697
training set, 761, 773, 1041
transformation, 1024
transitional flow, 914
transpose of a matrix, 579, 671, 697
trees, 26
triangular matrix, 569
trig functions, 391
trouble shooting libraries, 117
true accuracy, 43
true negative, 43
true positive, 43
true positive rate (*TPR*), 1048
turbulent flow, 914, 919
two-commodity model, 378
types, 98

U-substitution, 484
UC Irvine Machine Learning Repository, 156
UN Sustainable Development Goals (*UN SDG*), 1012
undetermined coefficients, 796
uniform distribution, 398, 401, 444, 450
uniform flow, 984, 998, 1005
universal sentence encoder (*USE*), 1039

variables, 98
variance and covariance, 726
vector equations, 556
vectorizations, 1025
vectors, 22
velocity potential, 982
velocity profile, 916
violin plot, 197
viscous flow, 914

water-flow rate, 912
web scraping, 1053
while loops, 100
word clouds, 54, 80, 88
words, 1017
World Health Organization (*WHO*), 903

zero matrix, 569
zero row, 552
zero vector, 556